NAVAL POSTGRADUATE SCHOOL
Monterey, California

1

THESIS

SELECTION AND INTEGRATION OF A GLOBAL POSITIONING SYSTEM RECEIVER WITH A CPU

by

Eric Twite

June, 1994

Thesis Advisor: Michael Shields

Approved for public release; distribution is unlimited.

Unclassified
Security Classification of this page

REPORT DOCUMENTATION PAGE					
1a Report Security Classification: Unclassified		1b Restrictive Markings			
2a Security Classification Authority		3 Distribution/Availability of Report Approved for public release; distribution is unlimited.			
2b Declassification/Downgrading Schedule					
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)			
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol	7a Name of Monitoring Organization Naval Postgraduate School			
6c Address Monterey CA 93943-5000		7b Address Monterey CA 93943-5000			
8a Name of Funding/Sponsoring Organization	6b Office Symbol	9 Procurement Instrument Identification Number			
Address		10 Source of Funding Numbers			
		Program Element No	Project No	Task No	Work Unit Accession No
11 Title SELECTION AND INTEGRATION OF A GLOBAL POSITIONING SYSTEM RECEIVER WITH A CPU					
12 Personal Author Twite, Eric					
13a Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report June 1994		15 Page Count *** 152	
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
17 Cosati Codes Field / Group / Subgroup	18 Subject Terms Global Positioning System, GPS, Differential Global Positioning System, Differential GPS, DGPS				

19 Abstract

The Global Positioning System (GPS) is revolutionizing the science of navigation. Never before has there been a system that could provide real time, world wide, continuous coverage with the such precision. Yet, the accuracy achievable with GPS alone is not sufficient to achieve autonomous flight of an Unmanned Aerial Vehicle. However, when integrated with an Inertial Navigation System and other non-inertial sensors using a Kalman Filter, GPS supplies the critical positioning information to permi such an achievement.

This thesis presents the selection and integration of a GPS receiver using Differential GPS (DGPS) in support of a UAV autonomous flight project. Contemporary electronic navigation systems are surveyed, GPS operation is reviewed, and a Motorola PVT-6 GPS receiver selected. Using the Motorola Proprietary Binary Format protocol, several software drivers were written in C to interface the information to an Intel 80486DX CPU using the RS-232 serial communication standard. Finally, an examination i made to determine the maximum reacquisition time, the DGPS accuracies achievable and the effects of pseudorange correction latency on DGPS accuracy.

20 Distribution/Availability of Abstract X unclassified/unlimited _ same as report _ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Michael Shields	22b Telephone (408) 656-2979	22c Office Symbol EC / SL

DD FORM 1473,84 MAR 83 APR edition may be used until exhausted security classification of this page
All other editions are obsolete Unclassified

Selection and Integration of a Global Positioning System Receiver with a CPU

by

Eric Twite
Lieutenant Commander , United States Navy
B.S., University of Southern California, 1983

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June, 1994

Author: Eric Twite

Eric Twite

Approved by: M. K. Shields

Michael Shields, Thesis Advisor

[illegible]

Isaac Kaminer, Second Reader

[illegible]

Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

The Global Positioning System (GPS) is revolutionizing the science of navigation. Never before has there been a system that could provide real time, world wide, continuous coverage with the such precision. Yet, the accuracy achievable with GPS alone is not sufficient to achieve autonomous flight of an Unmanned Aerial Vehicle. However, when integrated with an Inertial Navigation System and other non-inertial sensors using a Kalman Filter, GPS supplies the critical positioning information to permit such an achievement.

This thesis presents the selection and integration of a GPS receiver using Differential GPS (DGPS) in support of a UAV autonomous flight project. Contemporary electronic navigation systems are surveyed, GPS operation is reviewed, and a Motorola PVT-6 GPS receiver selected. Using the Motorola Proprietary Binary Format protocol, several software drivers were written in C to interface the information to an Intel 80486DX CPU using the RS-232 serial communication standard. Finally, an examination is made to determine the maximum reacquisition time, the DGPS accuracies achievable and the effects of pseudorange correction latency on DGPS accuracy.

TABLE OF CONTENTS

I. INTRODUCTION

The last of a 24 satellite constellation was launched into orbit in June 1993, and with it, "the science of navigation is poised to take a great leap forward" (Carey, 1993, p. B1). The Global Positioning System (GPS) revolutionized electronic navigation by providing worldwide, real-time, continuous coverage with three dimensional accuracies at least an order of magnitude better than any other system in existence. Even with these capabilities, it is not the panacea some make it out to be. Absolute accuracies (i.e., accuracies using a single receiver) better than ten meters within two standard deviations cannot be achieved.

Differential GPS (DGPS) shows great promise by further improving accuracies to within one to three meters. DGPS uses two GPS receivers where one is placed in a known location and the other is allowed to move. By computing the difference between the stationary receiver's actual and GPS determined locations, and then transmitting these differences to the other GPS receiver, the other GPS receiver can correct its output to obtain the improved position information. This improvement would be sufficient for an aircraft to achieve an autonomous landing with GPS alone were it not for the unreliability of the GPS signals in high speed, critical maneuvering evolutions. It is not uncommon for a GPS receiver to lose track of one or more of its satellites for a several seconds, and thereby, be unable to provide positioning information. While not a catastrophe for a slow moving surface vehicle, an outage of even a few seconds could prove fatal for an aircraft.

In an attempt to achieve the accuracy and reliability required for autonomous flight in a navigation package light enough for a small unmanned aerial vehicle (UAV), this research effort is part of a project where the GPS information is combined with the outputs of an Inertial Measurement Unit (IMU) and with data from other non-inertial navigation sensors such as airspeed and altitude indicators in a Kalman Filter. A contribution is made by selecting the best GPS receiver, defining its method of employment and writing the necessary software drivers to smoothly interface them with a Central Processing Unit (CPU).

In the following chapter, contemporary electronic navigation systems are surveyed, discussing their advantages and disadvantages. Realizing the obvious benefits of GPS, Chapter III examines the User Segment of GPS emphasizing those details necessary to intelligently select and use the GPS receiver in a manner that best supports the various project requirements. Chapter IV brings it all together by detailing the receiver selection and capabilities as well as the criteria for its use. It also describes the hardware and software necessary to interface the receivers with the CPU.

Three tests were conducted to confirm expected operational capabilities. They are described and the results from them are presented in Chapter V. The appendices include receiver specifications, the ephemeris algorithm and the software drivers written in C code as well as other information to aid the user and the programmer.

II. SURVEY OF ELECTRONIC NAVIGATION METHODS

The invention of wireless communications in the early twentieth century ushered in a new era in navigation allowing seafarers to safely pilot their ships in the absence of sight and sound for the first time in the history of mankind. The first use of radio waves required a vessel to transmit a continuous wave at some frequency. On shore, receivers with a rudimentary direction finding capability triangulated the ship's position and transmitted it back to the ship. The position was not very accurate by today's standards but served its purpose well in regions of the world where the sky was commonly obscured by clouds or fog when far from land.

The next significant advance came with the invention of highly accurate chronometers. First, the quartz clock, and then, the atomic time standards with mind boggling stabilities providing accuracies better than one second in 30,000 years or one part in a trillion (10^{-12}) (Logsdon, 1992, p.156). Such precision is required to allow the precise measurement of frequency. In addition, Transit (SatNav) and Global Positioning System (GPS) partially determine position by measuring the amount of time for the signal to travel from the satellite to the receiver. Radio waves travelling at the speed of light will advance one foot in one billionth of a second

(Kayton, 1990, p.336). It is not difficult to see the necessity for such accuracy in time keeping.

In order to appreciate the quantum improvement in the science of navigation that GPS has achieved, it is worthwhile to review the major electronic navigation methods in use today. A brief review of some principles and characteristics of electromagnetic waves is given, followed by an individual description of each of the major navigation systems. While an effort has been made to succinctly describe the theory upon which the system operates, the important point is to notice the range, coverage and accuracy for each method.

A. PRINCIPLES AND CHARACTERISTICS OF ELECTROMAGNETIC WAVES

Radio waves in a vacuum theoretically travel at the speed of light. Wavelength is related to frequency by Equation (1):

$$\lambda = \frac{2.997925 \times 10^8 \, m/sec}{f} \qquad (1)$$

where λ is the wavelength in meters and f is the frequency in Hertz. Figure 1 demonstrates this relation for the electromagnetic spectrum. Information may be carried on a radio wave through one or more modifications to its shape. These modifications include changing the amplitude, frequency or phase of the radio wave and are appropriately called amplitude, frequency and phase modulation, respectively. In addition, a wave may be intermittently transmitted or pulsed.

This is called pulsed modulation. Figure 2 shows three representative examples of modulation techniques.

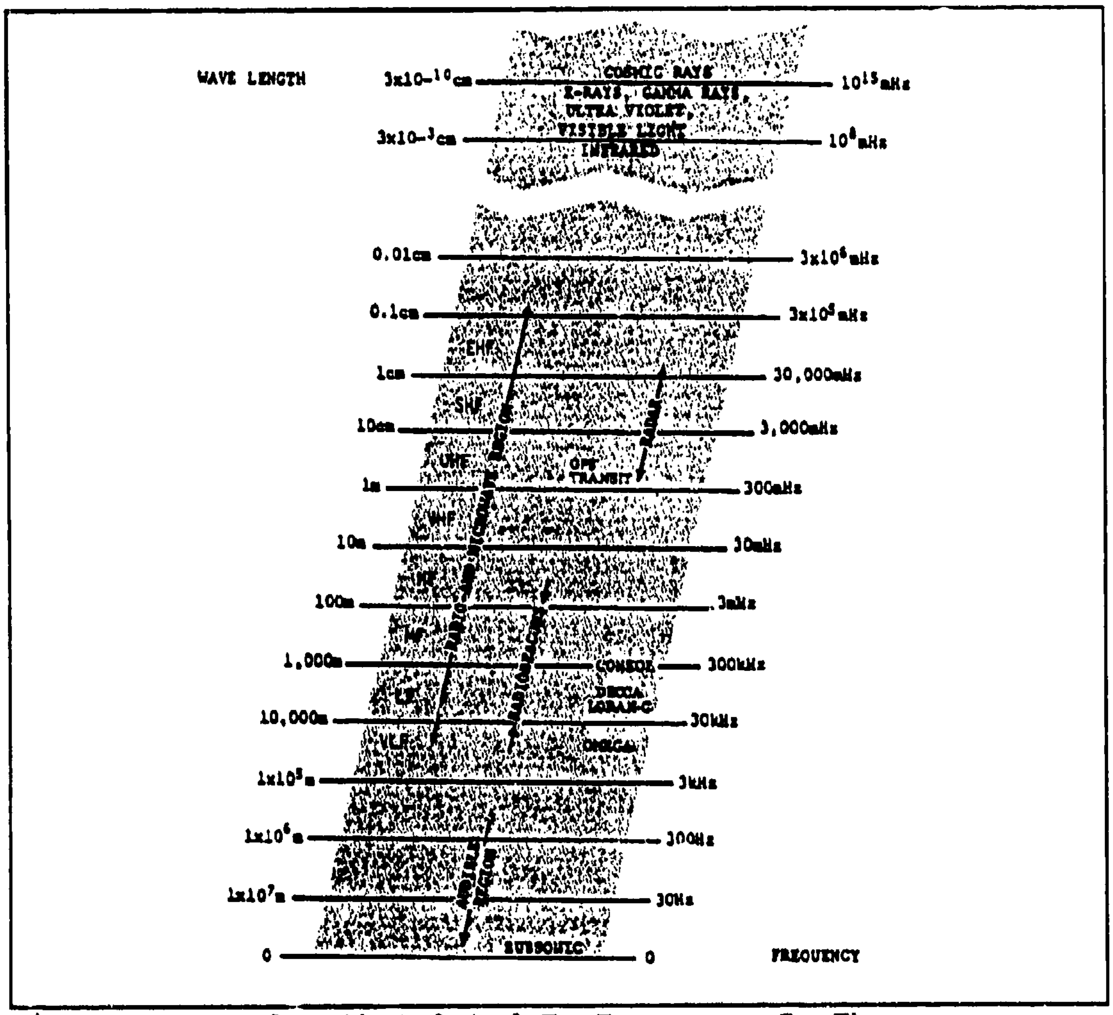


Figure 1. Wavelength Related To Frequency In The Electromagnetic Spectrum. (Hobbs, 1981, p. 217)

Radio waves, like light, experience such phenomena as reflection, absorption, refraction, diffraction and interference. With respect to the earth's environment, the two most influential features on the distance and path radio waves travel are the earth's surface and the ionosphere. The

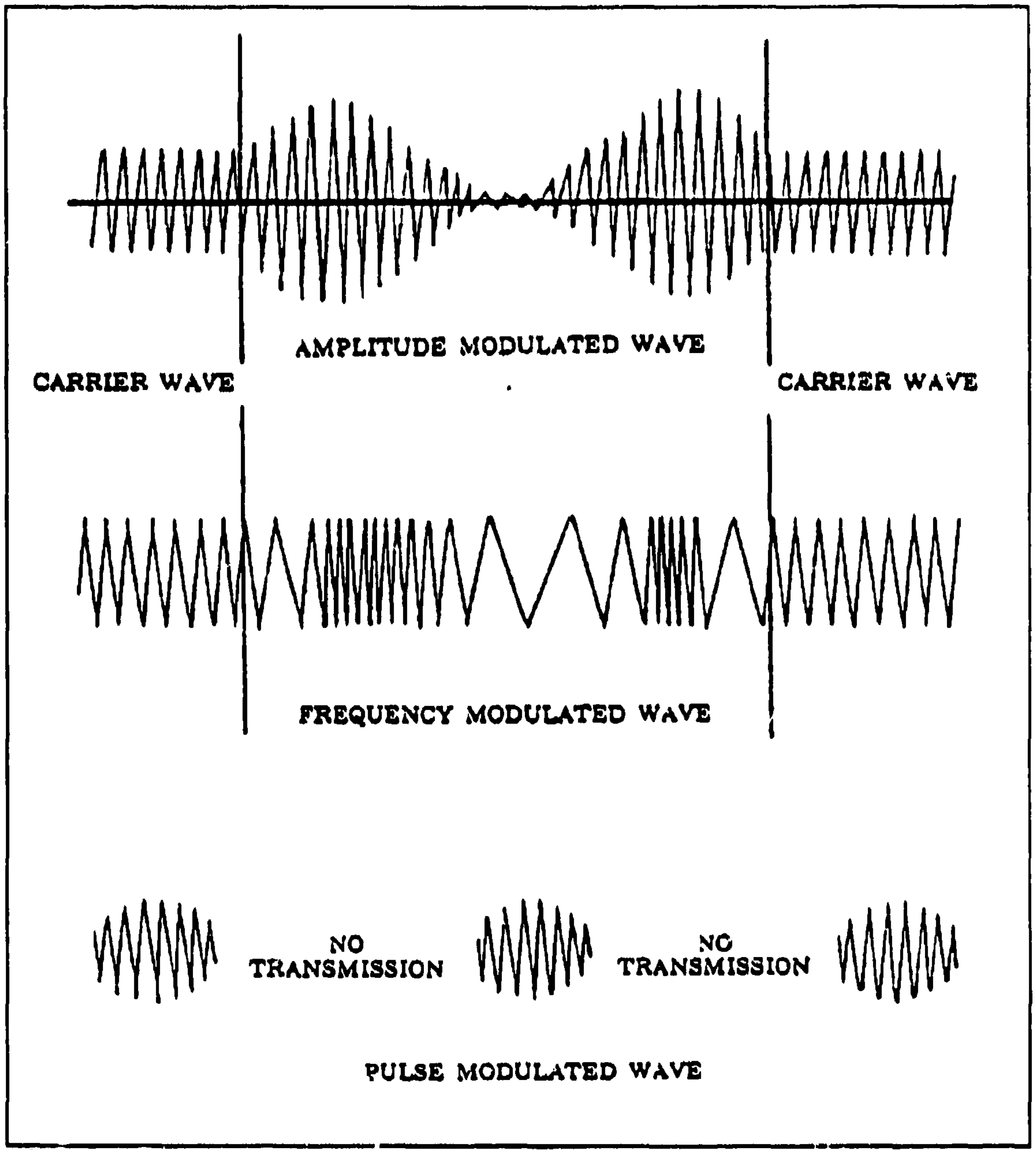


Figure 2. Modulation Techniques Commonly Used. (Hobbs, 1981, p.219)

ionosphere has four layers and exists at a height of approximately 60 to 400 Km above the earth's surface. This highly charged and fluctuating segment of the atmosphere not only absorbs radio wave energy but also reflects it (Beck, 1971, p.95). Waves that are reflected by the ionosphere are called "sky waves". The ionosphere is affected by diurnal and seasonal fluctuations and by the level of sunspot activity which in turn affects the performance characteristics of sky waves. Sky wave phenomena is also greatly influenced by the frequency, angle of incidence, and the height and density of the various layers of the ionosphere. The sky wave frequencies most commonly observed are in the 300 KHz to 30 MHz band (Medium and High Frequency bands). Because of the higher resistance of the earth's crust compared to the atmosphere, radio waves transmitted parallel to the earth's surface are slowed causing the wave to bend. The degree of bending is inversely proportional to the frequency. Thus lower frequencies will travel over the horizon in a direct path. These radio waves are called ground waves. As it might be expected, a receiver might travel along the surface of the earth from the transmitter origin and receive the ground wave, enter into an area shadowed by the earth's curvature with no signal, and then pick up the reflected "one-hop" sky wave even further out. Figure 3 illustrates this. Furthermore, when conditions are right, one-hop sky waves may experience reflection by the earth's surface and subsequent ionospheric reflection for a "two-hop" sky wave. Navigation systems such as Loran-C and Omega use reflected sky waves (when they exist) to extend their effective range. (Logsdon, 1992, pp. 95-98)

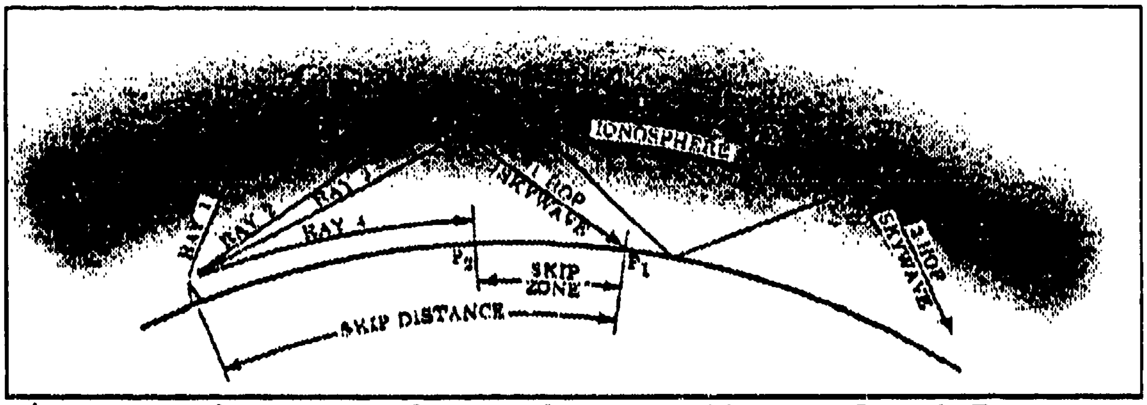


Figure 3. Sky Wave And Ground Wave Patterns. Ray 1 Is Refracted Through The Ionosphere, Ray 2 Is A One-Hop Skywave, Ray 3 Is A Two-Hop Skywave And Ray 4 Is A Ground Wave. (Hobbs, 1981, p. 225)

Finally, several major navigation systems exploit radio wave phase measurement in order to determine two dimensional position. The principle of operation uses two synchronized transmitters located a fixed distance apart. The line between them is called the baseline. As a result, at half-wavelength intervals the phase difference between the two signals is zero. In this type of system, the half-wavelength segments are normally called lanes. As the distance from the baseline increases the lines of equal phase or "isophase lines" radiate outward from the foci of the two transmitters in hyperbolic fashion as shown in Figure 4. Thus, methods employing this technique are called hyperbolic radio navigation systems. If the two transmitters were to transmit a pulse instead of a continuous wave, then by measuring the time of arrival of each synchronized pulse, one's position might also be determined. The points where the times of arrival of transmitted pulses are the same describe a hyperbolic locus. This type of method is also considered a hyperbolic radio navigation system. With the exception of radio beacons, all contemporary terrestrial radio navigation systems use some sort of hyperbolic navigation method. (Bowditch, 1984, pp. 1015-1016)

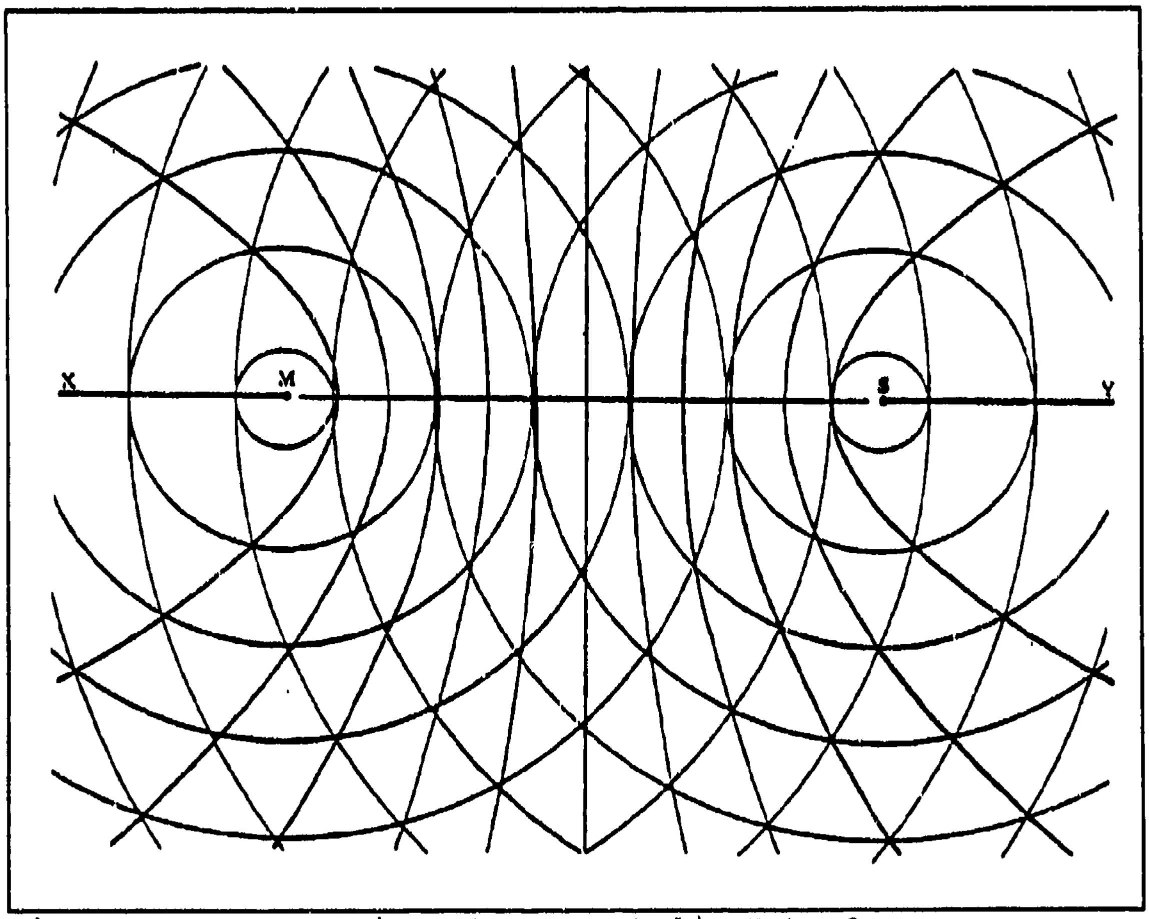


Figure 4. Construction Of A Hyperbolic Interference Navigation Pattern. (Hobbs, 1981, p. 227)

B. RADIO BEACONS

Radio beacons are continuous wave transmitters modulated with a unique Morse code identification. They are usually low power transmissions in the low and mid frequency bands with a range of less than 200 miles. Most cannot be received beyond

20 miles (Hobbs, 1981, p. 231). They operate using shipboard radio direction finding (RDF) equipment where two or more beacons are used to obtain a fix. The accuracy is highly dependent on the sensitivity of the RDF equipment as well as the range from the beacons. As Figure 5 illustrates, the geometry of the radio beacons relative to the ship may also play a significant role in the accuracy. Finally, the precise positions of the beacon transmitters must be known or the line of bearing is useless. In other words, the position information cannot be determined in terms of latitude or longitude until after it is plotted.

C. LORAN-C

Loran-C is the descendent of the first attempt at a hyperbolic navigation system developed during World War II for both ships and aircraft (Wilkes, 1987, pp. 31-34). From Figure 6, it can be seen that it covers almost the entire northern hemisphere with only a very small area covered below the equator and then only by sky waves when they exist.

Loran-C operates in the low frequency band at 100 KHz with a 10 KHz bandwidth. Within geographic areas, there is one master station and two or more secondary stations. All stations sequentially transmit a pulse modulated continuous wave in groups of eight pulses separated by one microsecond with the exception of the master station which transmits nine pulses. The ninth pulse is separated from the first eight by

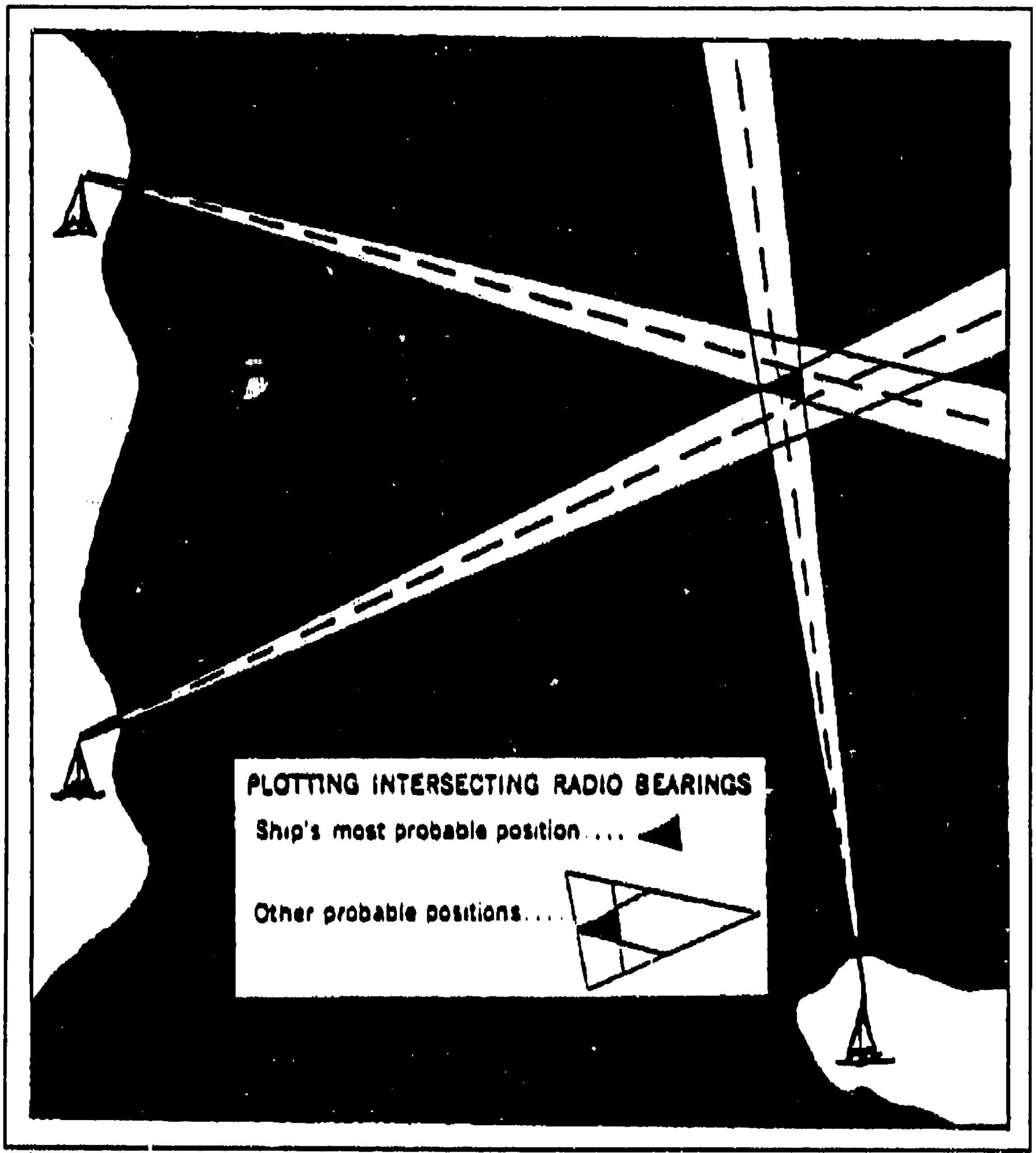

PLOTTING INTERSECTING RADIO BEARINGS
Ship's most probable position ....
Other probable positions....

Figure 5. Intersection Of Radio Beacon Signals Showing Error Spread As Range From The Source Increases. (Hobbs, 1981, p.235)

two microseconds and is used primarily for visual identification when viewing the pulses on an oscilloscope. This can be seen in Figure 7. Each group of master and

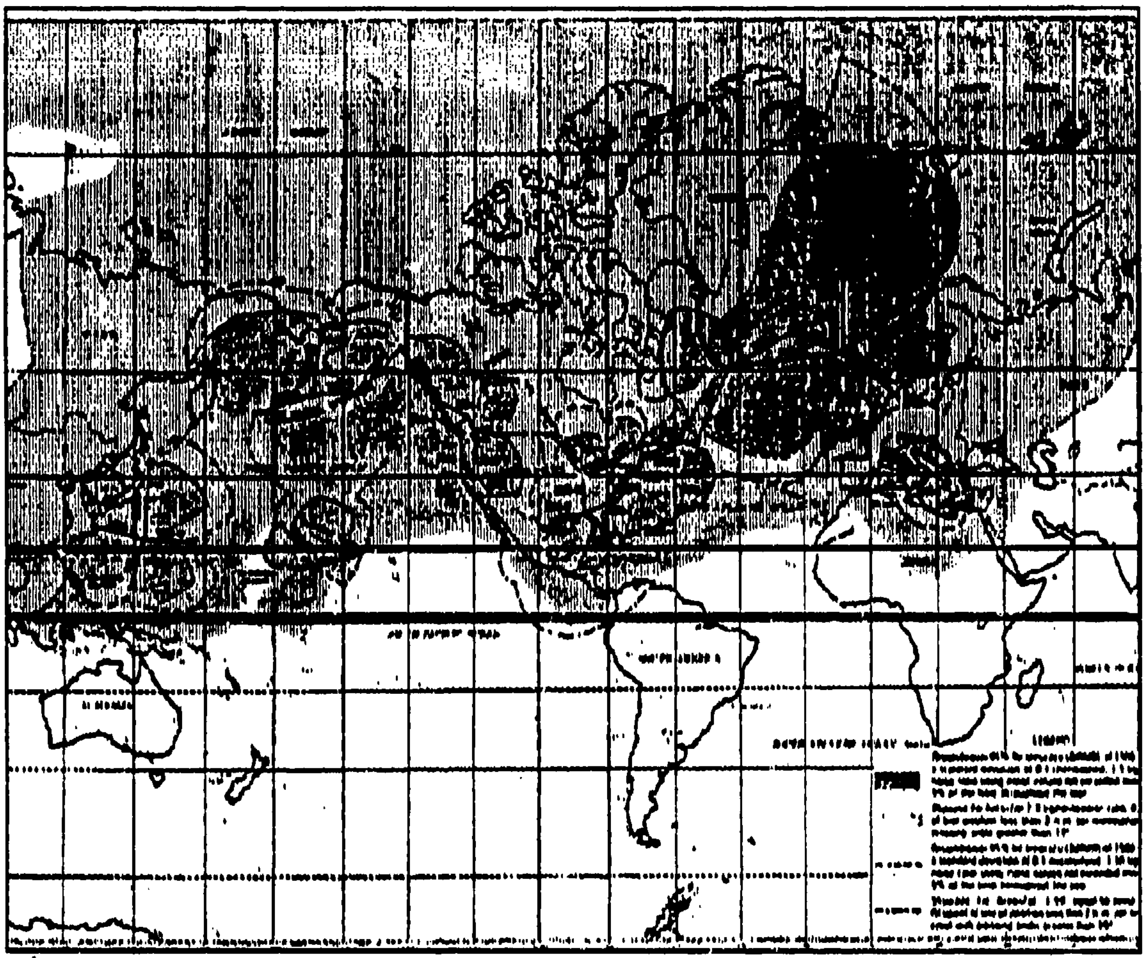


Figure 6. Areas Of The World Covered By Loran-C. In Particular, Note That The Coverage Is Almost Completely In The Northern Hemisphere. (Hobbs, 1981, p. 237)

secondary stations is called a chain. Currently, there exist 13 chains throughout the world. (USNO, 1972, p. 7-5) (Hobbs, 1981, pp. 236-239)

Each master and secondary station transmits according to a predetermined fixed order synchronized using precise atomic time standards. Within each receiver, the group of eight pulses are integrated to form one pulse of 320 microseconds. The time difference of arrival is then measured between the

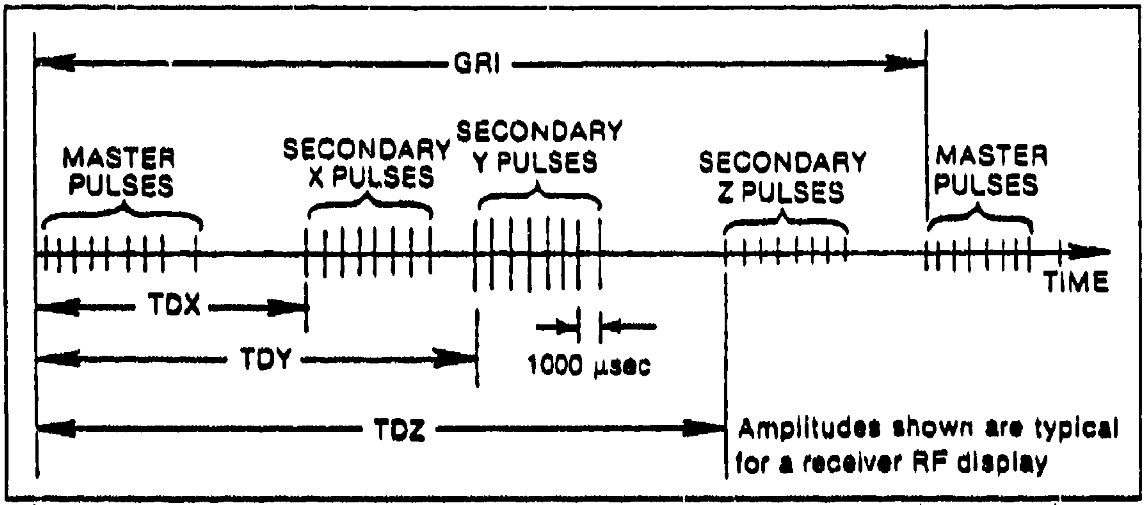


Figure 7. Loran-C Pulse Sequence For A Four Station Chain. (Hobbs, 1981, p. 239)

master and selected secondary station pulse from which may be plotted a line of position on Loran-C charts. As shown in Figure 8, Loran-C charts have the number coded hyperbolic lines overlaid on charts and to enable plotting. Chain and station identification are achieved by measuring the pulse repetition rate (PRR) and the pulse repetition interval (PRI). (USNO, 1972, p. 7-5)(Hobbs, 1981, pp. 236-239)

Accuracy of the system varies from about 700 ft near the baseline to about 2000 ft near the extreme range of the system. Due to the low frequency, baseline distances range from 1000 to 1500 miles so that coverage within a chain includes a great deal of area. Similar to the use of radio beacons, Loran-C charts and tables are required for the signal information to be meaningful.

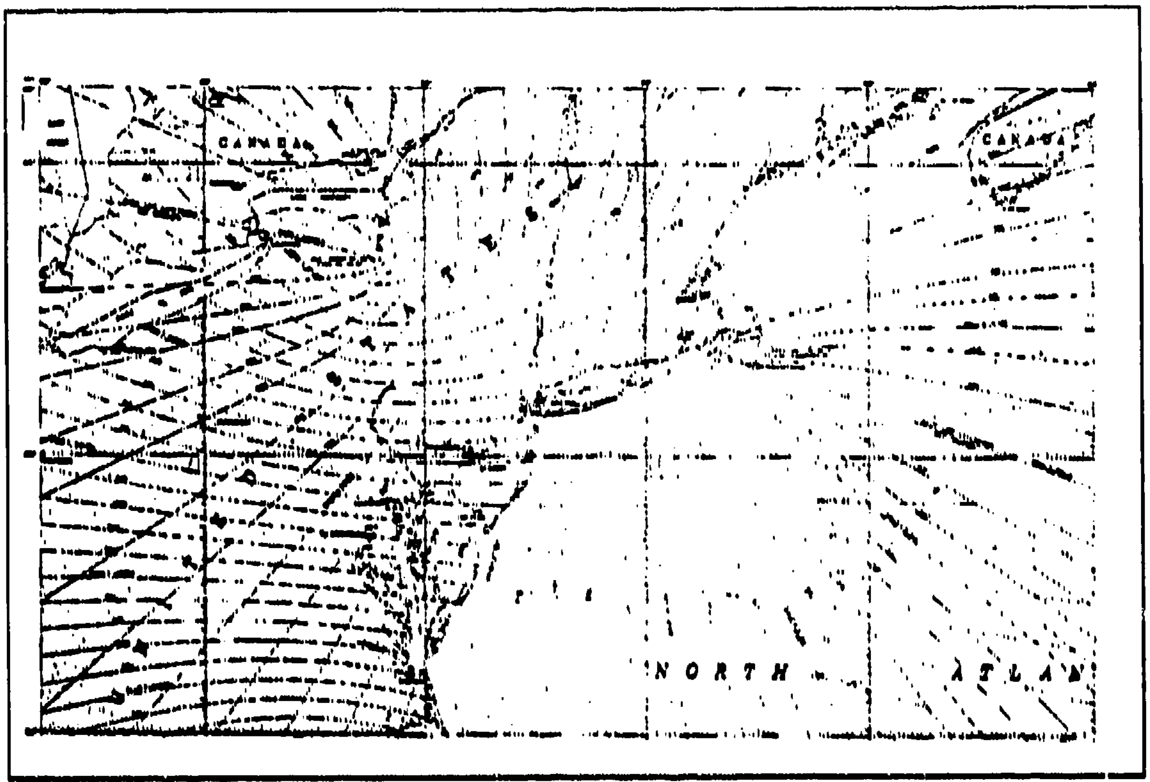


Figure 8. A Portion Of A Loran-C Chart Covered By The Northeast U.S. Chain. (GRI 9960) (Hobbs, 1981, 242)

D. DECCA NAVIGATOR SYSTEM

Decca was conceived by an American and developed by the British Admiralty Signals Establishment during World War II for use in mine sweeping and the Normandy invasion forces in 1944 (Beck, 1971, p. 65). Today, it is operated by the Racal-Decca Navigator Company in only a few but very heavily traveled areas in the world as shown in Figure 9. It operates using the principles of hyperbolic radio navigation but on a much smaller scale than Loran-C.

Each separate area of coverage has a master station and up to three slave stations arranged in a star pattern around the

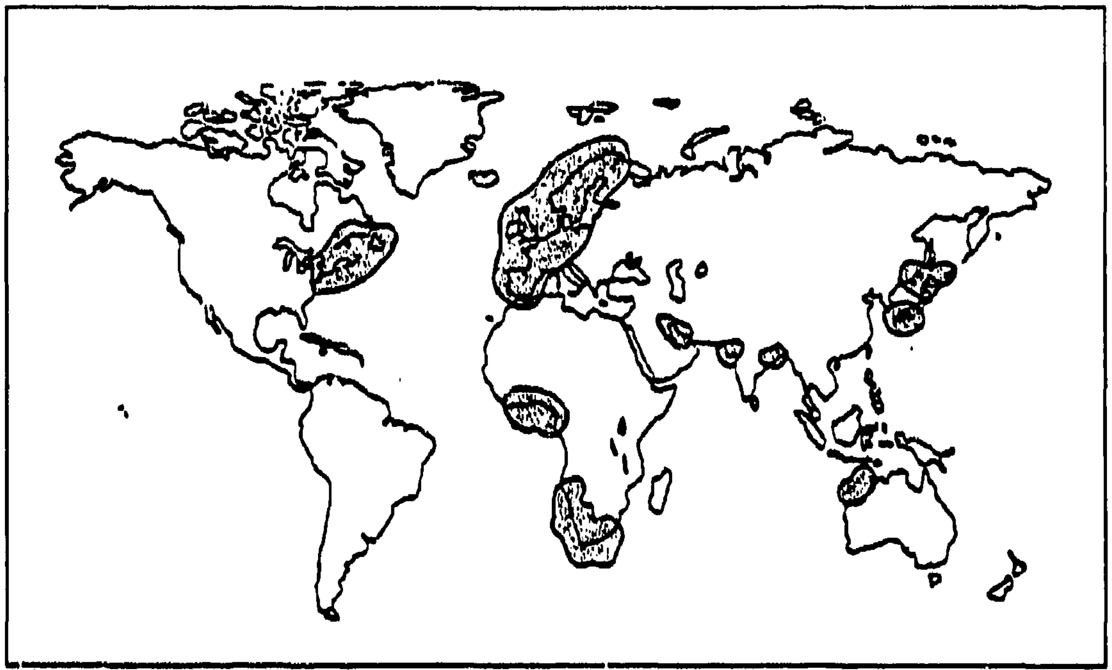


Figure 9. Areas Of The World Covered By Decca. (Beck, 1971, p. 64)

master station and identified by a color designator of green, red or purple. Each station within master/slave chain transmits an integer multiple of an unmodulated, continuous wave base frequency (f) in the range of 14.00 KHz to 14.33 KHz. The master station transmits at $6f$, the purple station at $5f$, the red station at $8f$ and the green station at $9f$. Each master/slave pair is phase locked along the baseline. A Decca receiver consists of four receivers, one for each of the frequencies transmitted by the master and slave stations. (Beck, 1971, pp. 65-67)

Within the receiver, each master/color frequency is integer multiplied to get another unique multiple of the base frequency which is the first common multiple of either of the two. For instance, the master station frequency of $6f$ when multiplied by four and the red station frequency of $8f$ when multiplied by three produce a new harmonic of $24f$ for the master/red station pair. Each half wave-length of this new harmonic is considered a "Decca lane", and by comparing phases of the two signals, one can determine position within the lane as Figure 10 illustrates.

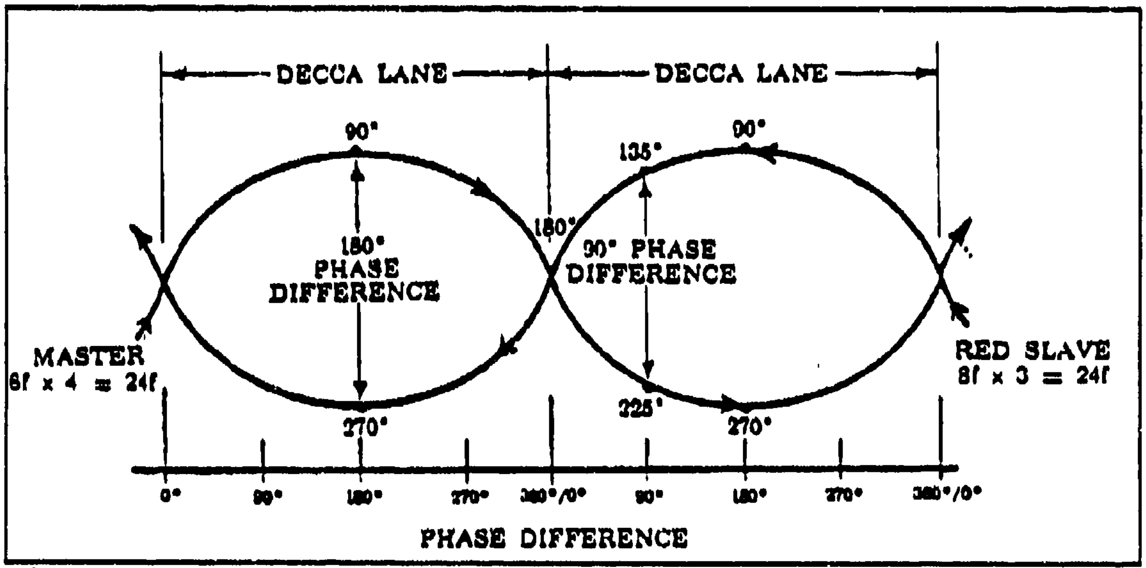


Figure 10. Phase Comparison Of Decca Master/Red-Slave Signals Along The Base Line. (Hobbs, 1981, p. 251)

Lane ambiguities are resolved once each twenty seconds when all four stations transmit the fundamental frequency f whose half-wavelength precisely encompasses the common multiples ($18f$ for the master/green pair, $24f$ for the master/red pair and $30f$ for the master/purple pair). By using phase comparison of the received signal in each of these "Decca zones" as shown in Figure 11, the exact lane can be determined. In a like manner, zone ambiguities are resolved by transmitting on a frequency of 8.2f also every 20 seconds. Called an orange frequency, it forms a coarse hyperbolic pattern within which each 360° difference cycle encompasses five zones. (Hobbs, 1981, pp. 252-253)

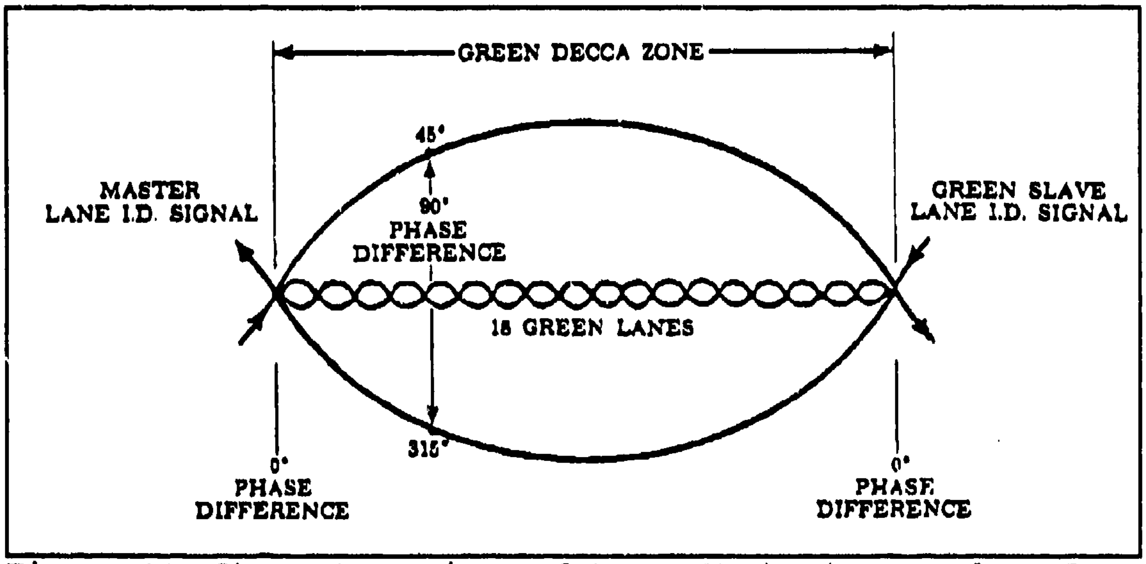


Figure 11. Phase Comparison Of Decca Master/Green-Slave Lane Identification Signals. (Hobbs, 1981, p. 252)

The effective range of each chain is approximately 240 miles from the master station. At 100 miles, the accuracy is approximately 30 yds by day and 100 yds by night. Again, this system requires the Decca charts and receiver in order to provide positioning data. (Bowditch, 1984, pp.1058-1059)

E. CONSOL

Consol was developed in 1945 from a German radio beacon system called Sonne. It is a hyperbolic radio navigation system with a very short baseline such that a "degenerated" hyperbolic pattern is formed. The curved portions of the hyperbolic pattern are not used, and so it acts as a long range radio beacon. It is used mostly in Europe and the Mediterranean. Another system very similar to Consol, called Cosolan, is used in the United States. Both have the advantage that no special equipment is required other than a low frequency receiver. The operator only needs to listen to the number of dots or dashes to know where within a particular sector the receiver is located. (Beck, 1971, 113-116)

Both systems use low frequency radio waves in the range from 190 KHz to 350 KHz. Each station consists of three radio towers spaced three wavelengths apart. One of the towers transmits a continuous wave frequency while the other two transmit waves that undergo 180° phase shifts with respect to the CW wave during a keying cycle. All the signals are modulated with dots and dashes. This arrangement produces a series of sectors 10° to 15° wide as shown in Figure 12. The phase shift of the combined signal results in a variable number of dots and dashes being heard depending on the radial position within the sector. (Beck, 1971, 113-116)

Consol range can be up to 1500 miles but not less than 25 miles due to the compressed width between sectors close to the transmitters. Keeping in mind that only radial lines of position may be obtained from a Consol station, accuracies of less than a degree may be obtained at the maximum range improving as the receiver approaches the station. Other than knowledge of the station location and the dot and dash

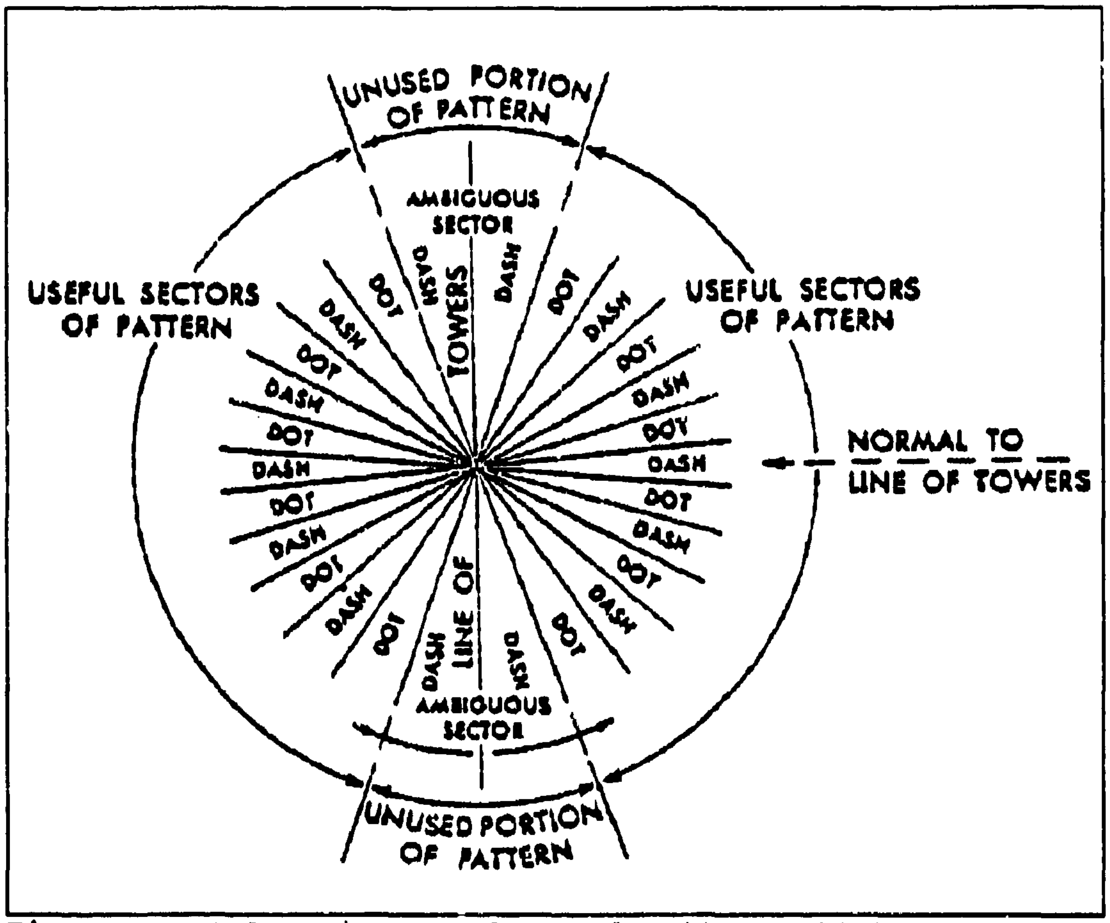

UNUSED PORTION OF PATTERN
AMBIGUOUS SECTOR
USEFUL SECTORS OF PATTERN
USEFUL SECTORS OF PATTERN
NORMAL TO LINE OF TOWERS
LINE OF TOWERS
DOT
DASH
AMBIGUOUS SECTOR
UNUSED PORTION OF PATTERN

Figure 12. Polar Diagram Of Consol Pattern. (USNO, 1972, p. 7-29)

position correlations, little is required to use Consol. (USNO, 1972, p. 7-5)

F. VOR/DME AND TACAN

VOR/DME stands for VHF Omni-directional Ranging/ Direction Measuring Equipment and TACAN stands for Tactical Air Navigation. It is, in effect, a radio beacon system for aircraft along heavily travelled airline routes and between military airfields and ships. The principle of operation of

TACAN is the same as for VOR/DME except that TACAN uses higher frequencies and is more accurate (Beck, 1971, pp. 26-28). Some VOR stations are equipped to respond to TACAN DME frequencies and are called VORTAC stations. VOR/DME uses VHF frequencies between 108 MHz and 118 MHz and TACAN uses UHF frequencies between 960 MHz and 1,215 MHz (Logsdon, 1992, p. 36). Consequently, both systems are line of sight only.

VOR employs two different signals working together in partnership: a narrow-beam rotating "lighthouse" transmission coupled with an "blinking" omni-directional pulse. As the lighthouse transmission sweeps past magnetic north, the blinking pulse is transmitted. Receiver equipment marks the time of receipt for the first pulse and then the time of receipt for the lighthouse sweep. Since the lighthouse sweep rotates at a fixed 30 rev/sec, it is a simple matter to compute the bearing from magnetic north. VOR provides only bearing information. (Logsdon, 1992, pp. 36-38)

For range from the VOR station, the aircraft transmits a series of interrogation pulses to the DME at the VOR station which then transmits a reply a fixed time after receipt. Equipment on the aircraft computes the slant range by dividing the time from interrogation to the time of receipt of the DME response by two and multiplying by the speed of light. Figure 13 illustrates this operation. (Logsdon, 1992, pp. 36-38)

At the extreme range of receipt, VOR can be off by as much as three degrees. However, as the range to the transmitter decreases, the error is reduced. Overall, VOR/DME TACAN positioning accuracies range from approximately 200 to 600 feet (Logsdon, 1992, p. 36). The obvious advantage over a radio beacon system is ack of need for RDF equipment on board the aircraft.

G. OMEGA

The Omega navigation system was designed to provide worldwide coverage and to be accurate. It consists of only eight stations, each using a base frequency of 10.2 KHz, and transmitting at 10 Kw power. This makes it possible to receive at least three and usually four stations anywhere on earth with an accuracy of approximately one nautical mile. Omega's principle of operation is similar to Decca in that it is a hyperbolic radio navigation system that uses the phase comparison of two CW transmissions to obtain a line of position. It is different in that any two stations can be used to obtain a line of position and with special receiving equipment, range from the two stations for a fix. (Bowditch, 1984, pp. 1016-1036)

Two 10.2 KHz continuous waves transmitted exactly in phase but travelling in opposite directions produce a series of hyperbolic lanes called Omega lanes. Each lane is eight nautical miles wide which corresponds to the half-wavelength of the frequency. Phase comparison determines the receiver position within the lane which is expressed in terms of

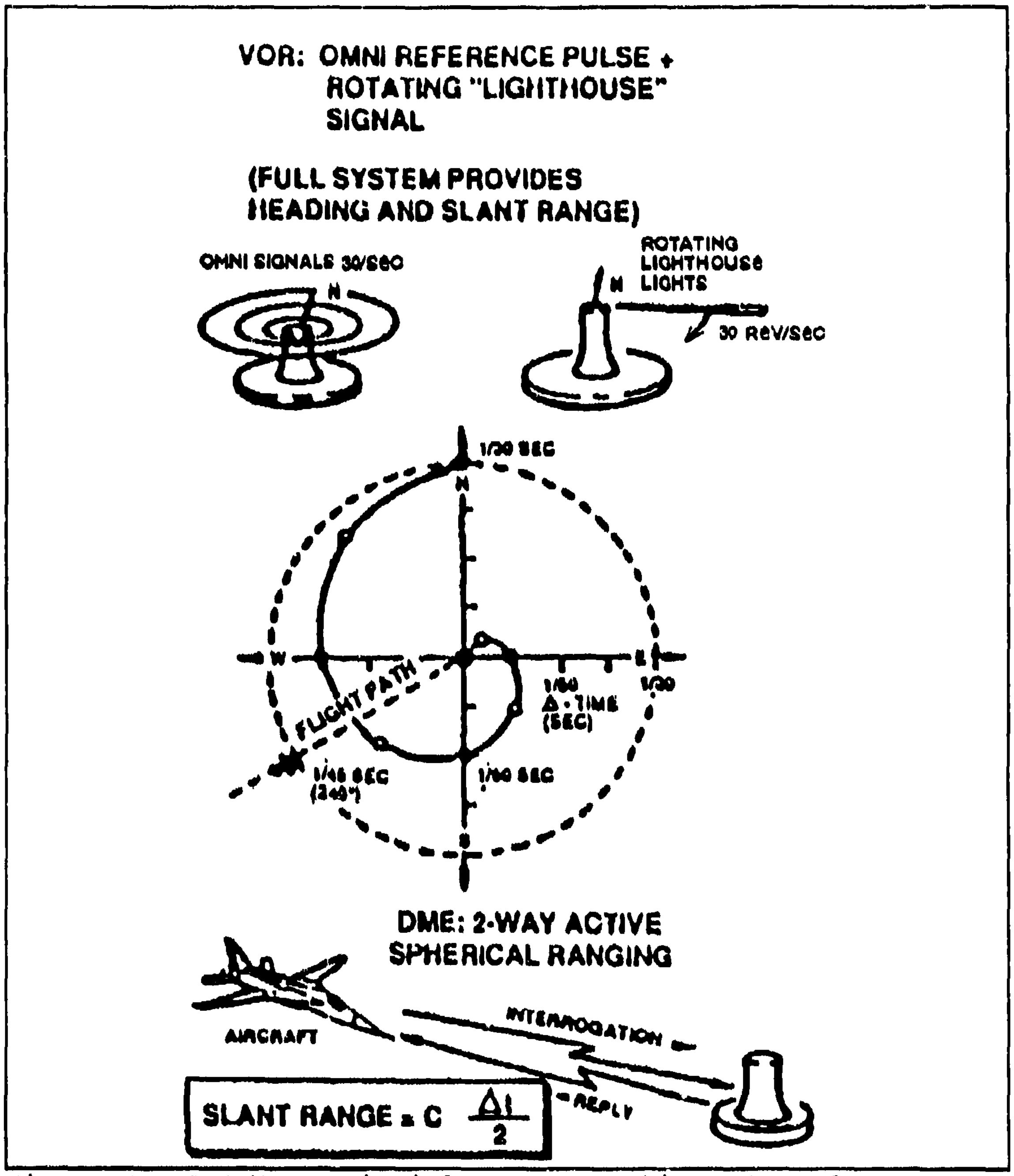

VOR: OMNI REFERENCE PULSE + ROTATING "LIGHTHOUSE" SIGNAL
(FULL SYSTEM PROVIDES HEADING AND SLANT RANGE)
OMNI SIGNALS 30/SEC
N
ROTATING LIGHTHOUSE LIGHTS
N
30 REV/SEC
1/30 SEC
N
W
E
1/60
Δ-TIME (SEC)
1/30
FLIGHT PATH
1/45 SEC (240°)
1/60 SEC
S
DME: 2-WAY ACTIVE SPHERICAL RANGING
AIRCRAFT
INTERROGATION
REPLY
SLANT RANGE = C Δt/2

Figure 13. VOR/DME Principles Of Operation. (Logsdon, 1992, p.37)

centicycles (cec) or centilanes (cel). These are defined to be 0.01 of the width of the lane. Figure 14 shows two Omega lanes produced by the 10.2 KHz frequency. (Hobbs, 1981, pp. 259-262)

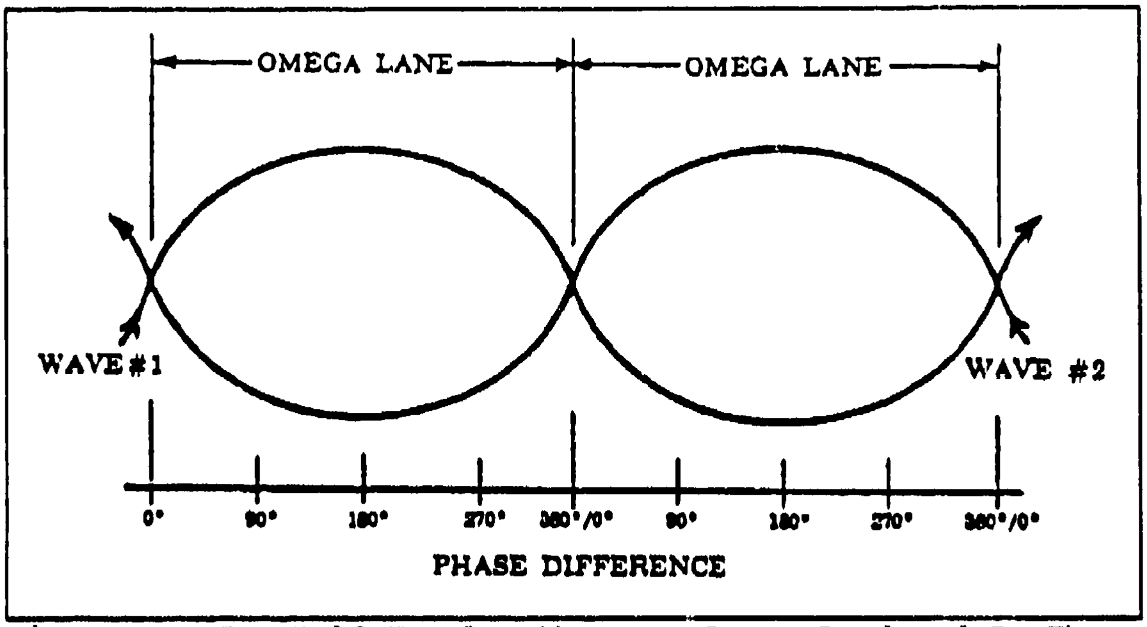
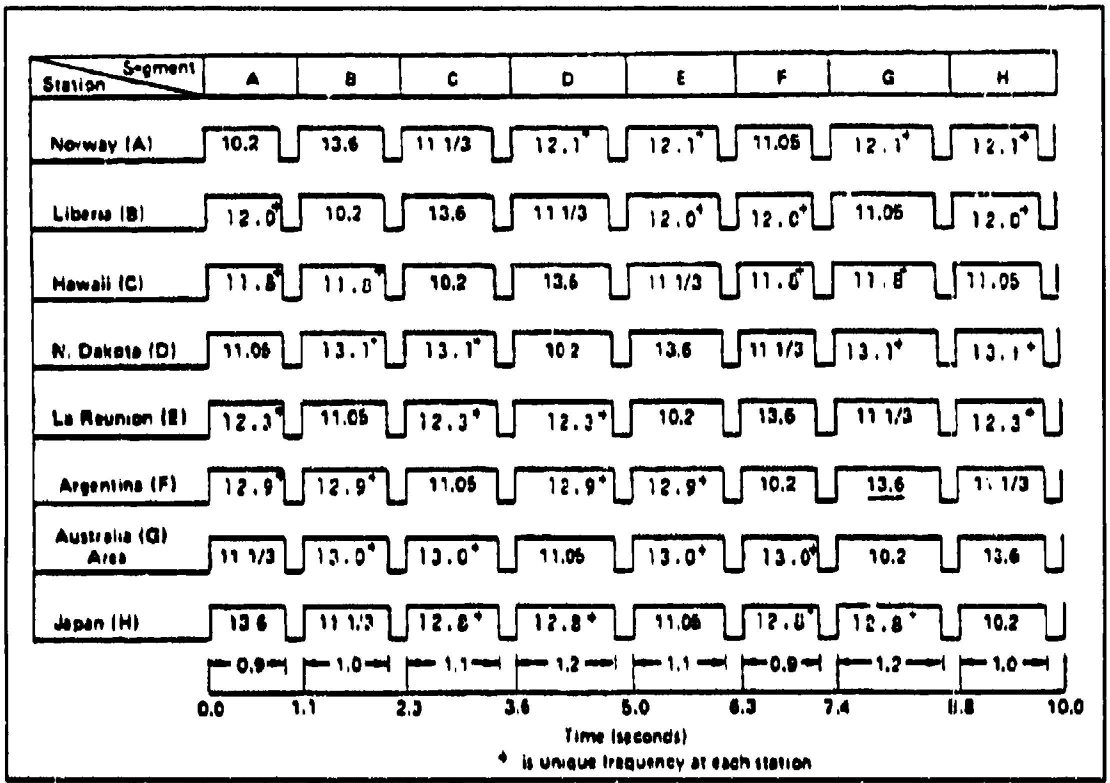


Figure 14. Two Half-Wavelength Omega Lanes Produced By The Phase Comparison Of The 10.2 KHz Signals. (Hobbs, 1981, p. 260)

Lane ambiguity is resolved by multiplexing the basic frequency with three others at 11.05 KHz, 11.33 KHz and 13.6 KHz over a ten second period according to the commutation pattern shown in Figure 15. Within the receiver, the difference frequencies of 0.283 KHz, 1.133 KHz and 3.4 KHz are extracted which are 1/36, 1/9 and 1/3, the frequency of the 10.2 KHz signal, respectively. The 3.4 KHz signal is used to establish a broader phase lane that encompasses three Omega lanes. Likewise, the 1.133 KHz signal phase lane encompasses three of the 3.4 KHz phase lanes and so on. By performing successive phase comparisons as demonstrated in Figure 16, the operator can easily determine within which Omega lane his receiver is located. (Hobbs, 1981, pp. 259-262)



Station \ Segment	A	B	C	D	E	F	G	H
Norway (A)	10.2	13.6	11 1/3	12.1*	12.1*	11.05	12.1*	12.1*
Liberia (B)	12.0*	10.2	13.6	11 1/3	12.0*	12.0*	11.05	12.0*
Hawaii (C)	11.8*	11.8*	10.2	13.6	11 1/3	11.8*	11.8*	11.05
N. Dakota (D)	11.05	13.1*	13.1*	10.2	13.6	11 1/3	13.1*	13.1*
La Reunion (E)	12.3*	11.05	12.3*	12.3*	10.2	13.6	11 1/3	12.3*
Argentina (F)	12.9*	12.9*	11.05	12.9*	12.9*	10.2	13.6	11 1/3
Australia (G) Area	11 1/3	13.0*	13.0*	11.05	13.0*	13.0*	10.2	13.6
Japan (H)	13.6	11 1/3	12.8*	12.8*	11.05	12.8*	12.8*	10.2
	0.9	1.0	1.1	1.2	1.1	0.9	1.2	1.0

0.0 1.1 2.3 3.6 5.0 6.3 7.4 8.8 10.0

Time (seconds)

* is unique frequency at each station

Figure 15. Omega Station Commutation Pattern. (Hobbs, 1981, p. 262)

H. SHIP'S INERTIAL NAVIGATION SYSTEM (SINS)

Although it requires no external input, SINS is still an extremely useful navigation system. It is found primarily on submarines and aircraft carriers. Inertial navigation is...

> "... the process of directing the movements of a vessel based on sensed accelerations in known spatial directions by means of instruments that mechanize the Newtonian laws of motion, integrating such accelerations to determine velocity and position." (Hobbs, 1981, p. 281)

In essence, an inertial navigation system such as SINS is a highly accurate dead reckoning computer.

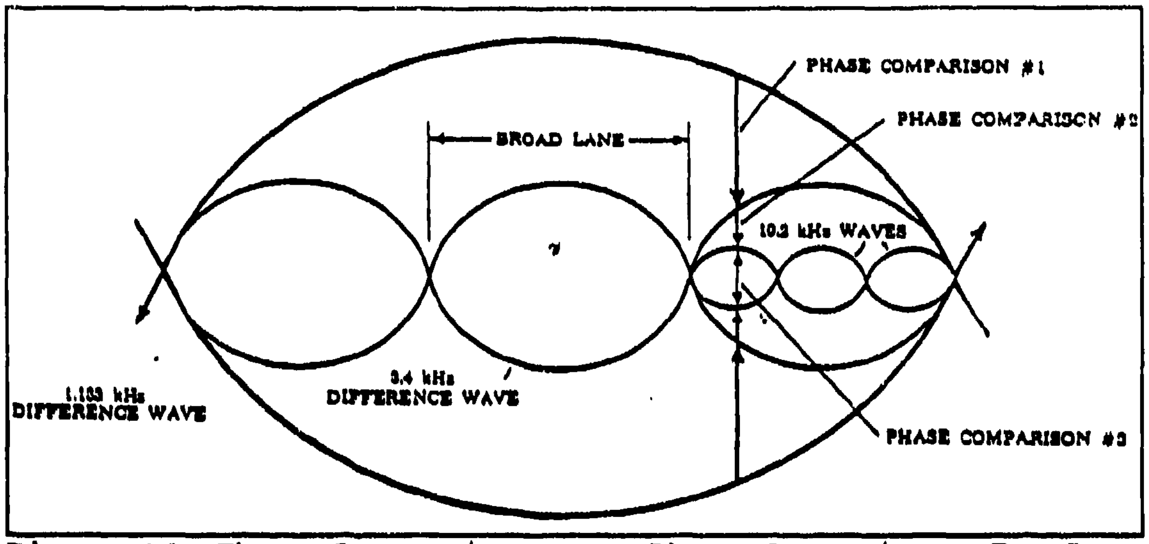


Figure 16. Three Successive Omega Phase Comparisons For Lane Resolution On Surface Ships. (Hobbs, 1981, p. 261)

SINS mounts two accelerometers on a platform stabilized by a system of three gyros in such a way that it is constantly coincident with a plane tangential to the earth's surface. The two accelerometers are continually oriented in a north-south and east-west direction; hence, they are sensitive only to horizontal north-south and east-west accelerations. Integrating the accelerations with respect to time gives the component velocities from which the ship's true velocity may be computed. (Hobbs, 1981, p.281-282)

Because of the cumulative affect of a number of possible sources for errors such as friction in the gyro supports and disturbances caused by the daily rotation of the earth, SINS accuracy degrades over time and must be updated from external sources. One improvement to SINS has been the Electro-Static Gyro (ESG) which is used to reset or update the SINS. Simply, the ESG is a one centimeter diameter beryllium sphere spinning at 216,000 rpm in a near perfect vacuum. This rotor is supported solely by an electrostatic field. The sphere is thus freed from the classical gyro bearing friction as well as many of the associated random torques that a mechanical support can introduce. The ESG extends the interim period between external updates by a factor of six. (Kayton, 1990, pp. 194-206)

Another improvement in SINS technology is the Ring Laser Gyro (RLG). It differs from the traditional concept of a gyro in that it contains no spinning mass. It is a closed geometric laser path (usually triangular such as the one in Figure 17) centered on an expected spin axis. Identically phased laser beams are continuously generated, which travel in opposite directions around the closed path. Any rotation about the spin axis causes an apparent phase difference in the two beams at a measurement point at one vertex of the triangle, proportional to the speed of rotation. When combined with the output of three mutually orthogonal accelerometers, position and velocity are easily computed. RLG's increase the precision over an ESG monitored SINS by an order of magnitude. (Hobbs, 1981, p.281-282)

Finally, in an effort to further improve accuracy and reduce size and weight, the latest development is the Fiber Optic Gyro. Its principle of operation the same as RLG's except the laser path can be tightly wound in a spool and mirrors are not required. The technology is very young and promising and is still being perfected.

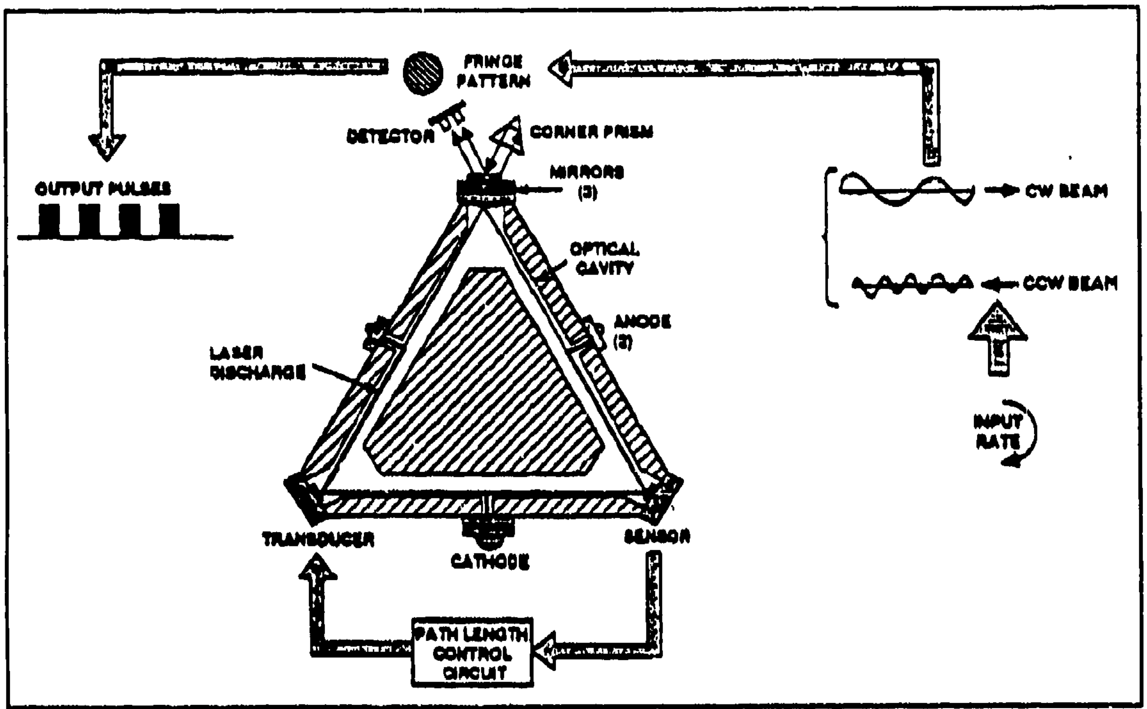


Figure 17. Single Axis Ring-Laser Gyro. By Monitoring The Interference Pattern Caused By The Two Counter Directional Isophase Light Beams, Extremely Small Rotations Can Be Measured. (Logsdon, 1992, p. 107)

I. SHIPBOARD DOPPLER SONAR SYSTEM

Another navigation system that does not provide absolute position as much as it provides accurate inputs to a dead reckoning system is the Shipboard Doppler Sonar System. Because of its extreme accuracy in measuring very slow velocities, it is found on very large ships where ship's speed over ground is important. For example, when mooring to a pier, the ship could crush the pier if the contact velocity were greater than a knot (Hobbs, 1981, p. 285).

It operates by measuring the doppler shift of two or more sonar beams in order to compute speed over ground. One such system by Raytheon, called Janus, uses two fixed hull mounted transducers that transmit pulsed sonar signals in two beams, one forward and the other aft, at about a 30° angle from the vertical. Figure 18 illustrates a similar configuration that uses four transducers. The former of the two produces fore and aft speed accuracies of 0.1 kt, depth accuracies to the nearest foot, meter or fathom and distance accuracies to the nearest hundredth of a nautical mile. (Kayton, 1990, pp. 188-193)

J. TRANSIT

Transit Navigation Satellite System, sometimes called just Transit or SatNav, can trace its conceptual origins directly to the launch of Sputnik I. It consists of five satellites in polar orbits at an altitude of 580 nm with periods of revolution of 107 minutes as shown in Figure 19. The rest of the system includes ground tracking stations, a computing center, an injection station and receivers. Its geometry is configured such that every satellite comes within range of every position on earth at least twice a day at 12 hour intervals. The average time between fix opportunities approximately 100 minutes at the equator and a half an hour near the poles. The difference is due to the intrinsic nature of polar orbits. (Bowditch, 1984, pp. 1066-1095)

Transit employs two frequencies in the UHF band of 150 MHz and 400 MHz. It operates by using an integrated doppler method

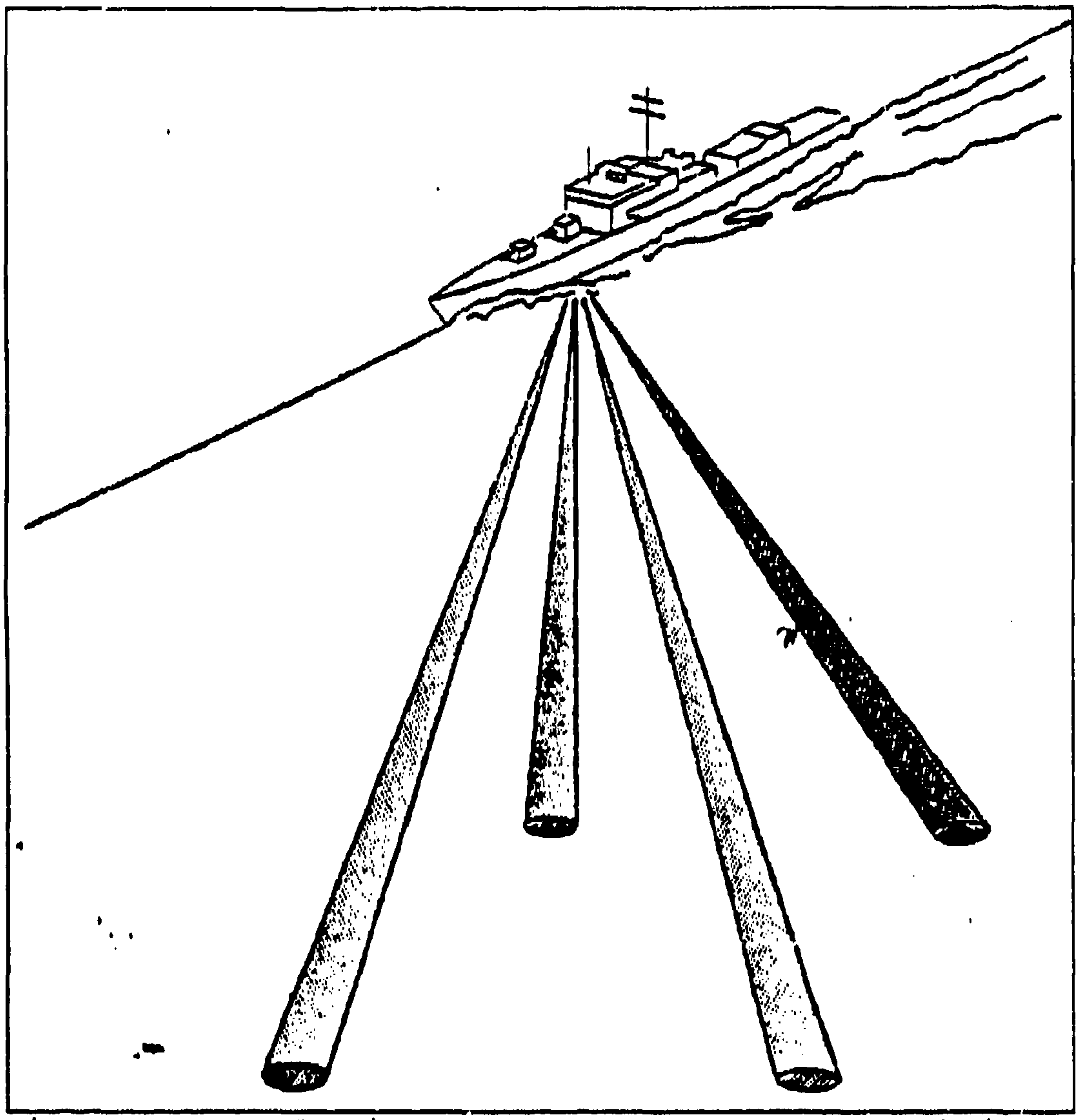


Figure 18. A Dual-Axis Four-Beam Janus Sonar Array Of The Raytheon Doppler Sonar System. (Hobbs, 1981, p. 285)

where the slant range from the receiver to the satellite is computed by fitting the doppler phase shift curve of the satellite to look up table. Each satellite transmits a phase modulated message that repeats every two minutes containing the satellite time, ephemeris and other information from which

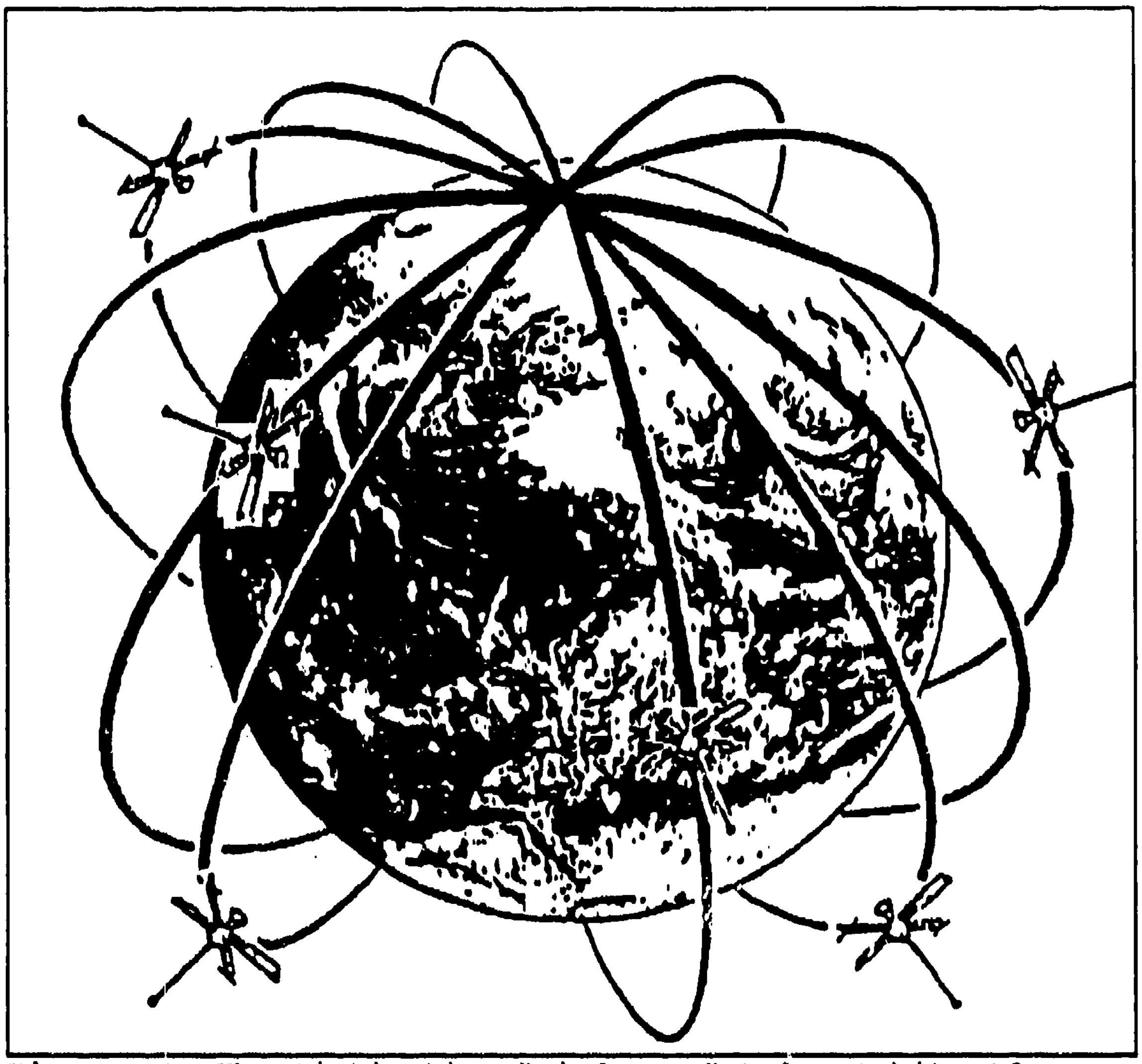


Figure 10. The Distinctive "Bird Cage" Polar Orbits Of Transit Satellite System. (Hobbs, 1981, p.275)

the receiver calculates the receivers position in space and computes the longitude and latitude. Within in each pass of the satellite, the receiver needs to collect at least three successive messages for a fix. Four are preferred and seven are optimal. The number receivable is dictated by the height above the horizon achieved by the satellite. (Bowditch, 1984, pp. 1066-1095)

The accuracy of Transit is affected by ships dead reckoning accuracy and speed of advance. For a stationary ship, it is common to achieve accuracies of 35 meters made possible by the shorter wavelength of the UHF signals (Bowditch, 1984, pp. 1067). It is significant to note that while this system is world wide and has improved positioning accuracies of better than an order of magnitude over systems discussed so far, it is comparable to Omega in its coverage and worse in its availability to provide positioning information.

K. GLOBAL POSITIONING SYSTEM (GPS)

With the June 1993 launch of the twenty-fourth satellite, NAVSTAR's Global Positioning System (GPS) became fully operational in late 1993. GPS provides real time three dimensional positioning and timing information anywhere in the world. It consists of 24 uniformly spaced satellites in six orbital planes at an altitude of 20,180 Km as depicted in Figure 20. Like Transit, it uses two UHF frequencies at 1,575 MHz and 1227 MHz simultaneously upon which constellation ephemeris data, atmospheric propagation correction data and satellite clock error are phase modulated. (Wells, 1987, pp. 4.00-4.11)

The principle of operation for GPS is to calculate the range from four different satellites using the broadcast message, and to compute the longitude, latitude and altitude

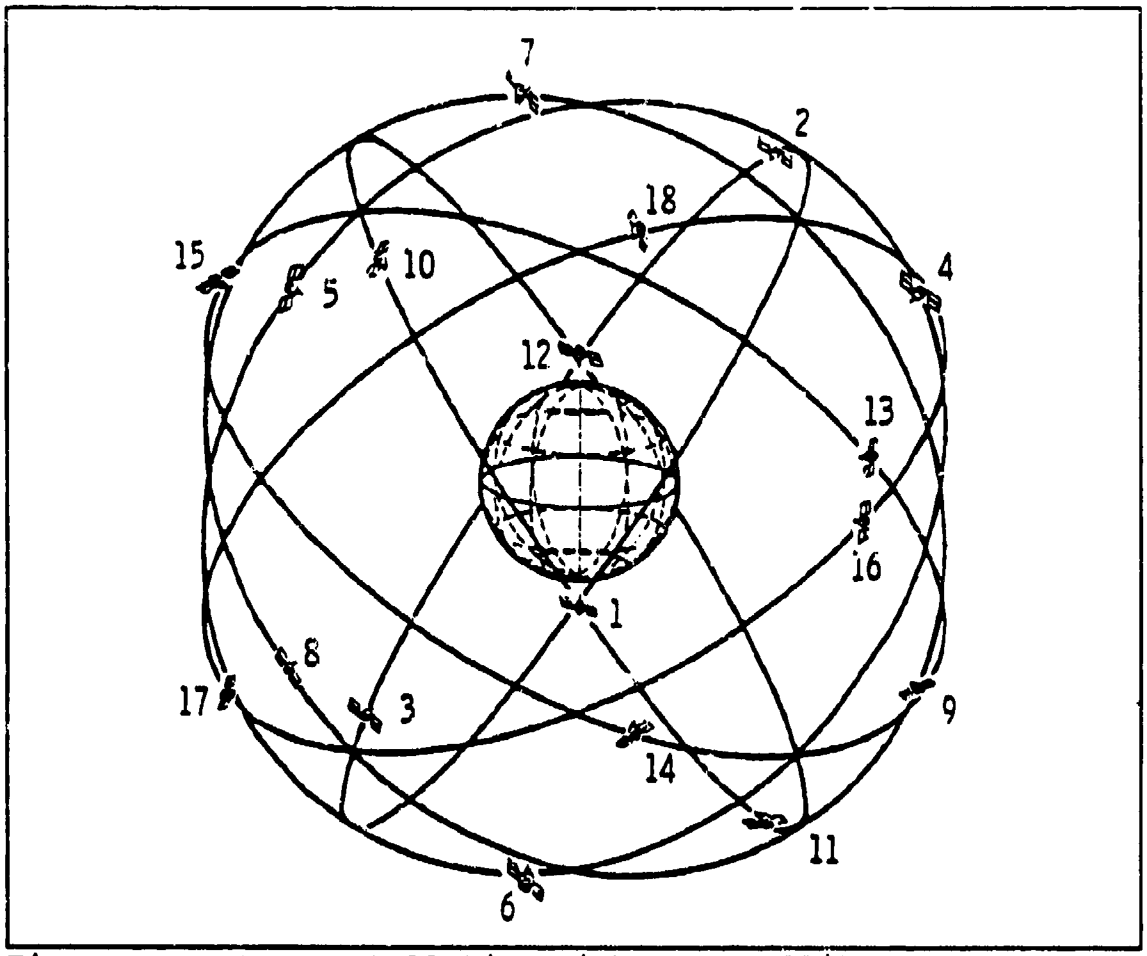


Figure 20. GPS Constellation With 24 Satellites Ensures At Least Twelve In View At All Times Anywhere In The World. (Hobbs, 1981, p. 288)

of the simultaneous solution of the resulting ranges. Because the system is operated by the Department of Defense for military use, the lower of the two frequencies is encrypted and altered to deny any potential enemy real time positioning accuracy for medium and long range targeting. Still, accuracies using just the higher frequency are approximately 100 meters (Logsdon, 1992, p.64). Using methods to be explained in the next chapter, this can be improved by more than an order of magnitude and in non-real time applications such as surveying, mind boggling accuracies of a millimeter spherical error probable (SEP) can be achieved using phase comparison techniques (Logsdon, 1992, p.64).

L. SUMMARY OF CONTEMPORARY ELECTRONIC NAVIGATION SYSTEMS

It is easy to see why, after surveying the various electronic navigation systems available, GPS invites the most enthusiasm and excitement from all users. It is a quantum improvement over anything else available. There are other navigation systems not discussed here such as Microwave Landing Systems (MLS), the French Argos system and the Russian GPS imitation GLONASS. All suffer from one or more limitations in range, coverage, availability or accuracy. What is important in this review is an appreciation of the varicus contemporary electronic methods and their method of operation including their particular limitations. In the next chapter, the GPS principle of operation is more closely examined as well as the various methods for its use.

III. GLOBAL POSITIONING SYSTEM

GPS operational and technical characteristics must be understood in detail in order to follow the reasoning for GPS receiver selection and the concept of employment in interfacing them with the CPU. In the following sections, a comprehensive review of the User Segment of GPS is provided. It covers the salient points of signals, coding, message content, message structure, calculations, and various biases. The chapter ends by using this information in a description and discussion of Differential GPS.

GPS is functionally divided into three segments: Control segment, Space segment and User segment. The Control segment exists to monitor, update and operate the whole system. The Space segment consists of the 24 satellites. These two critically important segments, however, are of no concern to the user and little is required to be known of them. The User segment is the most important, for understanding its operation will greatly influence the user's choice of receiver and subsequent method of employment.

A. USER SEGMENT RECEIVER HARDWARE DESCRIPTION

Currently, receivers cost as little as a few hundred dollars to as much as $50,000 or more. Size and weight also vary greatly from receivers as small as a package of cigarettes weighing approximately the same to larger than a car battery and weighing only slightly less. Shape and type of antenna can also impact the effectiveness of the receiver. Figure 21 illustrates the most common designs of antennae found on the market.

One type of antenna called a microstrip is relatively thin (approximately 1/10" to 1/2" thick) and can be designed to receive one or both of the frequencies that GPS satellites transmit. This design is ideal for use in this project where low drag is desired on an aircraft. However, signal reception from satellites low on the horizon usually experience a much lower signal-to-noise ratio due to the lower gain of the antenna at those low angles caused by its thin design.

Other factors that distinguish receivers from one another are the choice of position solution algorithm and interface communication protocols. All of the above characteristics are determined and limited by the state of the art of the technology and by the purpose for which it was designed.

Receivers may be loosely grouped into four broad categories based on capability (Logsdon, 1992, p. 66):

1. Number of channels and sequencing rate.
2. Access to selective availability signals.
3. Use of available performance enhancement techniques.
4. Computer processing capabilities.

The majority of user receiver sets sold are continuous tracking or parallel receivers. They typically have from four

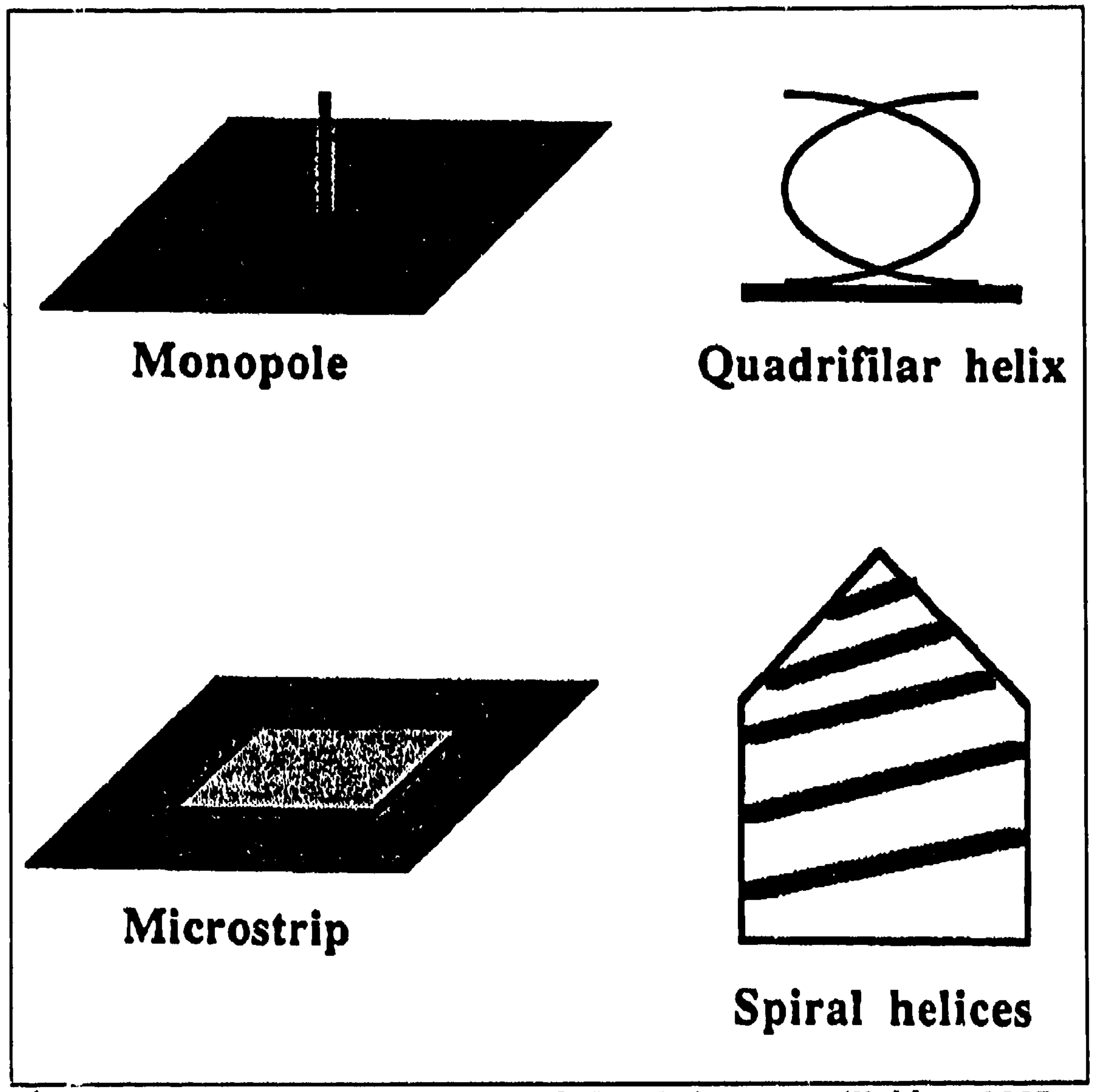


Figure 21. Four Common Types Of GPS Antennae. (Wells, 1987, p. 7.02)

to twelve channels. There are also single channel receivers capable of providing relatively crude positioning. To obtain data from multiple satellites, they employ one of three different sequencing speeds: slow, fast or multiplexed (very fast) as shown in Figure 22. (Logsdon, 1992, p. 66-67)

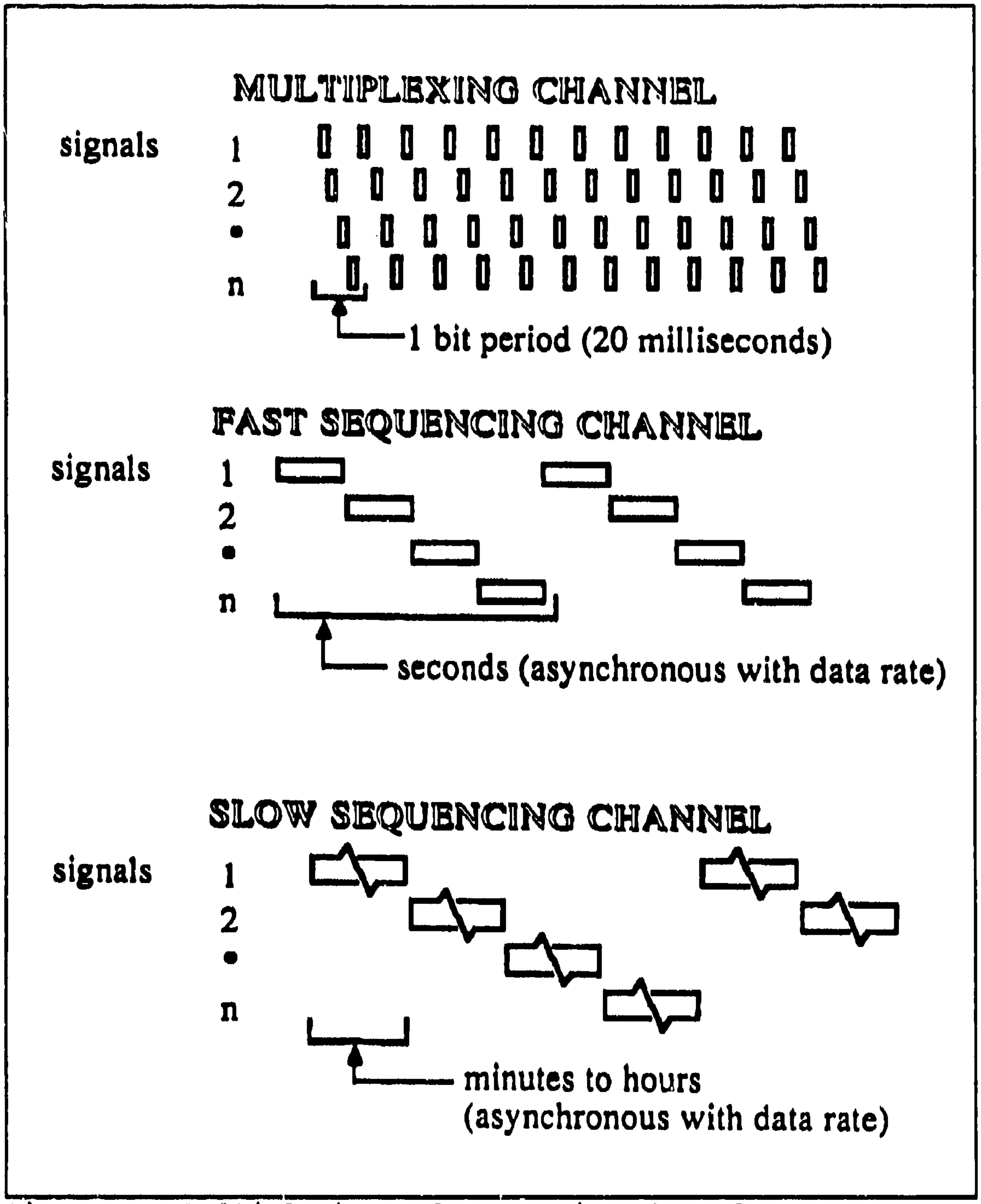


Figure 22. Multiplexing And Sequencing Channels For GPS Receivers. (Wells, 1987, p. 7.13)

Selective Availability (SA) is the primary means by which the United States Department of Defense denies the potential terrorist or enemy accurate use of GPS for targeting navigation for missiles. GPS messages are encoded using two codes. SA alters and encrypts the message content of the more accurate Precision (P) code carried on both frequencies. The less accurate Course and Acquisition (C/A) code message is called the Standard Positioning Service (SPS) and is available to anyone on the higher frequency only. The P-code information is called the Precise Positioning Service (PPS). Only those receivers with a decoding nip (tightly controlled by the DOD) can receive PPS information. (Logsdon, 1992, p. 67)

SPS receivers can still achieve excellent accuracies by using performance enhancement techniques. One such technique capable of centimeter accuracy uses interferometry. This process needs very expensive antennae and receivers, and the data has to be post-processed. Surveyors are the most common users with this accuracy requirement and do not need the real time positioning information so denial of PPS is not a significant handicap. Another capability that enhances SPS accuracy in near real-time uses a process called Differential GPS (DGPS). Simply, DGPS employs two receivers: one stationary at a known location and one mobile. The stationary receiver computes the difference between its known location and its GPS derived location. It then transmits those differences to the mobile receiver which uses them to correct its own GPS derived position. Since the Selective Availability errors are the same over baselines as large as 600 nm, accuracies approaching and exceeding those of PPS receivers can be achieved. DGPS is explained more completely later. (Logsdon, 1992, p. 67)

The last category of receivers is distinguished by computer processing capability. Mostly esoteric, this is where the manufacturer takes the GPS message and calculates the output message accounting for all the nuances of the solution algorithm. These include such items as whether or not to use just four satellites or more when they are available, whether or not to account for ionospheric and tropospheric delays (sometimes modeling these delays can introduce more errors than they eliminate), when to shift to other satellites that will reduce the geometrical error, etc. In addition, the receiver can be enhanced through the use of a Kalman Filter to help smooth the position and time data as well as reduce receiver processor time. (Logsdon, 1992, p. 68)

B. GPS SIGNALS AND CODING

GPS satellites continuously transmit two frequencies: L1 at 1575.42 Mhz and L2 at 1227.60 Mhz. The respective wavelengths for L1 and L2 are approximately 20 cm and 25 cm. Each of these is modulated with the GPS message at 50 bps. Since each satellite uses the same frequencies to send its own unique message, the receiver must be able to distinguish one signal from the other. Furthermore, the transmitting power of each satellite so is low that by the time the signal reaches the receiver, its signal-to-noise ratio is less than the ambient noise level. GPS overcomes these obstacles by using spread spectrum and code division-multiple access techniques. As mentioned earlier, the messages are encoded using two codes called C/A code and P-code. The chipping rate of the C/A code and P-code is 1.023 Mbps with a virtual wavelength of 290 m and 10.23 Mbps with a virtual wavelength of 29 m, respectively. Each satellite uses its own unique code with which the receiver may auto-correlate a duplicate and thus phase lock on to it. The combination of the transmission, coding and message chipping rates provides for a very robust signal with which it is very difficult to interfere. (Wells, 1987, pp. 4.5 - 4.10)

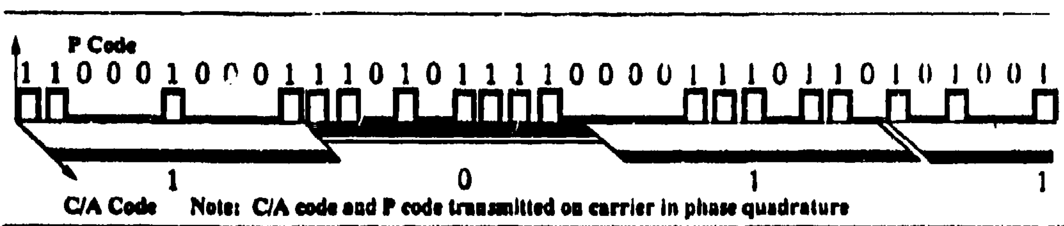
The C/A code uses a 1023 bit Gold code with a period of one millisecond. The Gold code was chosen because of its length and because of the property that Gold codes are nearly orthogonal. It is also very easy to replicate and with its short period, easy to achieve correlation with itself. A G signal can only be received if the code assignment is known ahead of time. The Gold code assigned to each satellite is published in the Interface Control Document *ICD-GPS-200* published by Rockwell Space Systems Division. (Wells, 1987, p. 6.07)

The P-code uses a much longer code with 235,469,592,765,000 bits and has a period of approximately 267 days. It is divided into 32 seven day segments, and a segment is assigned to each satellite. The DOD further encrypts the P-code to produce the Y-code to deny its use to unauthorized PPS users. The encryption and length of the P-code make it extremely difficult to correlate to without keying information and decryption hardware. (Wells, 1987, p. 6.07)

The GPS message is phase modulated onto its respective codes which in turn are phase modulated onto their respective carrier frequencies. As mentioned earlier, the L1 frequency is modulated with both the C/A code and the P-code. This is accomplished by phase modulating its quadrature component with the P-code. This feature primarily allows the military to employ PPS redundancy in certain applications. The table in Figure 23 compares the important points of the code and data features. (Wells, 1987, pp. 6.3 - 6.8)

C. GPS MESSAGE CONTENT

A GPS receiver determines its position by computing the position of four or more satellites within view and then measuring its range from each. By solving four simultaneous equations, the receiver calculates its coordinates in an Earth Centered Earth Fixed (ECEF) cartesian coordinate system. The system chosen is defined to be consistent with the World Geodetic Survey of 1984 (WGS-84) which places the origin at the intersection of the Z axis (through the earth's center of mass and parallel to the north pole of the Conventional Terrestrial Pole (CTP)) and the X-axis (the intersection of the WGS-84 Reference Meridian Plane parallel to the zero



Parameter	C/A Code	P Code	Data
CHIPPING RATE	1.023 X 10^6 bits/sec	10.23 X 10^6 bits/sec	50 bits/sec
SPATIAL LENGTH (PER BIT)	290m (960 ft)	29m (96 ft)	6000km (3235 n. m.)
REPETITION INTERVAL	0.001 sec	7 days	Not applicable
CODE TYPE	Gold code	267 day pseudo random code	Not applicable
TOTAL NO. OF CODES	37 unique Gold codes	37 seven-day sections	Not applicable
SPECIAL PROPERTIES	Easy to acquire	Slightly more accurate. Resistent to jamming and spoofing. Rejection of multipath.	Provides handover from C/A to P code, ephemeris data and clock correction.

Figure 23. GPS Code Comparison. (Applied Research Laboratories, 1993, p. 2.2)

meridian plane, and the plane of the CTP's equator). The Y-axis completes the ECEF right hand orthogonal coordinate system measured in the equatorial plane 90° to the east as shown in Figure 24 (Applied Research Laboratories, 1993, p. 2.1). Having defined the reference frame for GPS position, it is necessary to understand a few more concepts in describing position within an ECEF system.

The GPS receiver translates the ECEF coordinates into geodetic coordinates for the user. The equations to translate coordinates from ECEF to geodetic coordinates and vice-versa

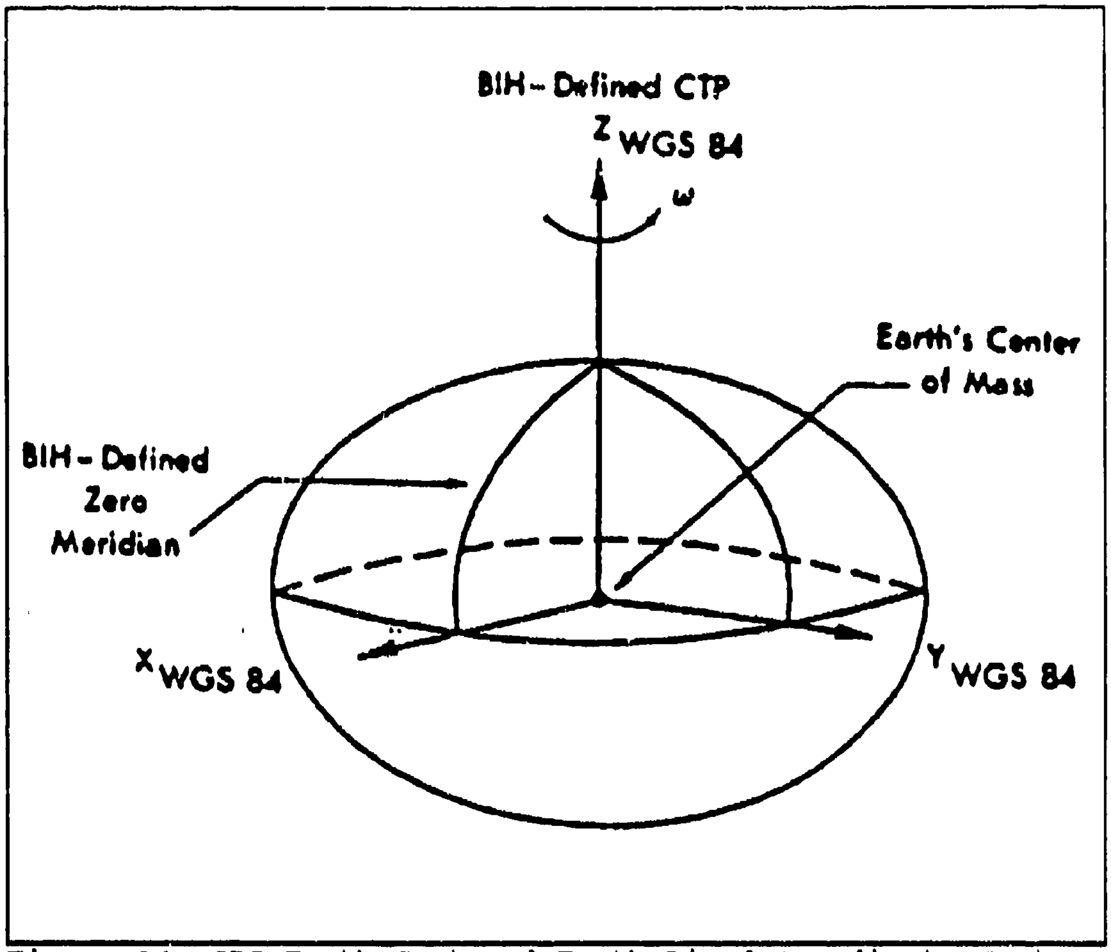


Figure 24. GPS Earth Centered Earth Fixed Coordinate System As Defined By The WGS 84. (BIH Stands For Bureau International De L'Heure) (Applied Research Laboratories, 1993, p. 2.1)

are provided in Appendix A. By using the reference ellipsoid model of the world defined by WGS 84, GPS is capable of relating pin point positions between continents. Before the World Geodetic Surveys of 1972 and 1984, inter-continent surveying was virtually impossible. Cartographers could only establish local datums to which all surveyed points were referenced. These datums were quite accurate for their respective geographical regions at best. One of the accomplishments of WGS 84 was in relating all the different local datums to the ECEF coordinate system. By modelling the datum as a point of tangency of an ellipsoid as shown in Figure 25, the ellipsoid origin could easily be referenced to the origin of the ECEF coordinate system. With the advent of WGS 84 and GPS, the importance of the local datums has been diminishing. Most GPS receivers contain a library of the major local ellipsoid references to the WGS ECEF coordinate system so charts and maps that use different local datums can be employed with GPS.

Once the reference frame is established and understood, satellite position can be described using the six classical Keplerian orbital elements. They are briefly described in Figure 26 but they are not sufficient for the most accurate description of the satellite orbit. This is because the predicted position of a satellite is the result of least-squares curve fitting which is not valid for an entire orbit. Thus, it is necessary to include several other parameters to more accurately model satellite positions in space and time. All the parameters are listed in Appendix B. All told, there are 17 specific parameters included in the GPS message.

D. GPS CALCULATIONS

The equations required to determine the ECEF coordinates of the satellites from the ephemeris are presented in Appendix

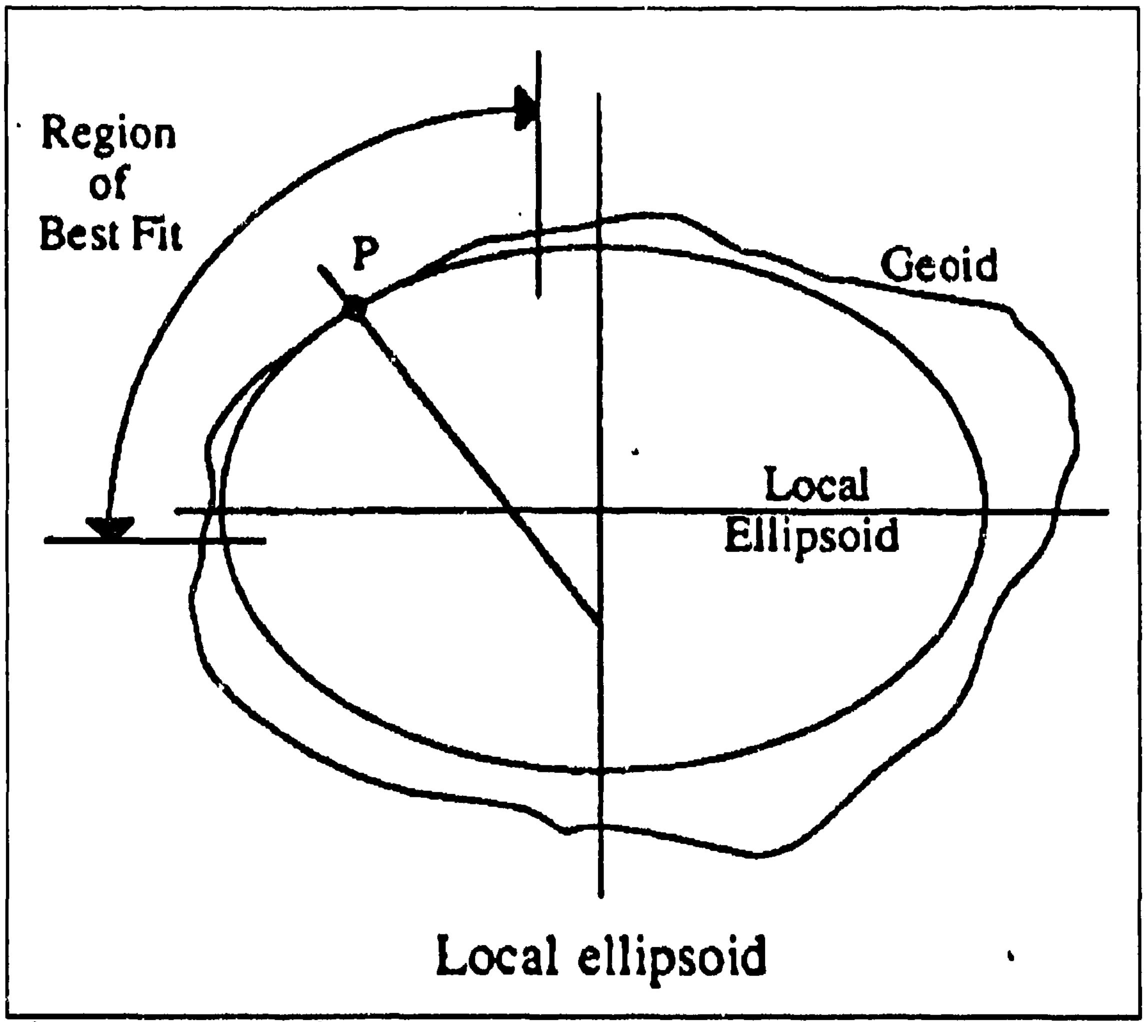


Figure 25. Two Dimensional Illustration Of A Theoretical Ellipsoid Tangent To The Geoid At An Arbitrary Datum P. (Leick, 1990, p. 189)

B. If the receivers and satellites could be perfectly synchronized with GPS time, three parameters would be unknown: X, Y, Z. It would then be possible to determine position using only three satellites. However, all receivers use inexpensive, lightweight and comparatively inaccurate quartz clocks and each has a bias from GPS time for which there must be an accounting. Therefore, for three dimensional position, there

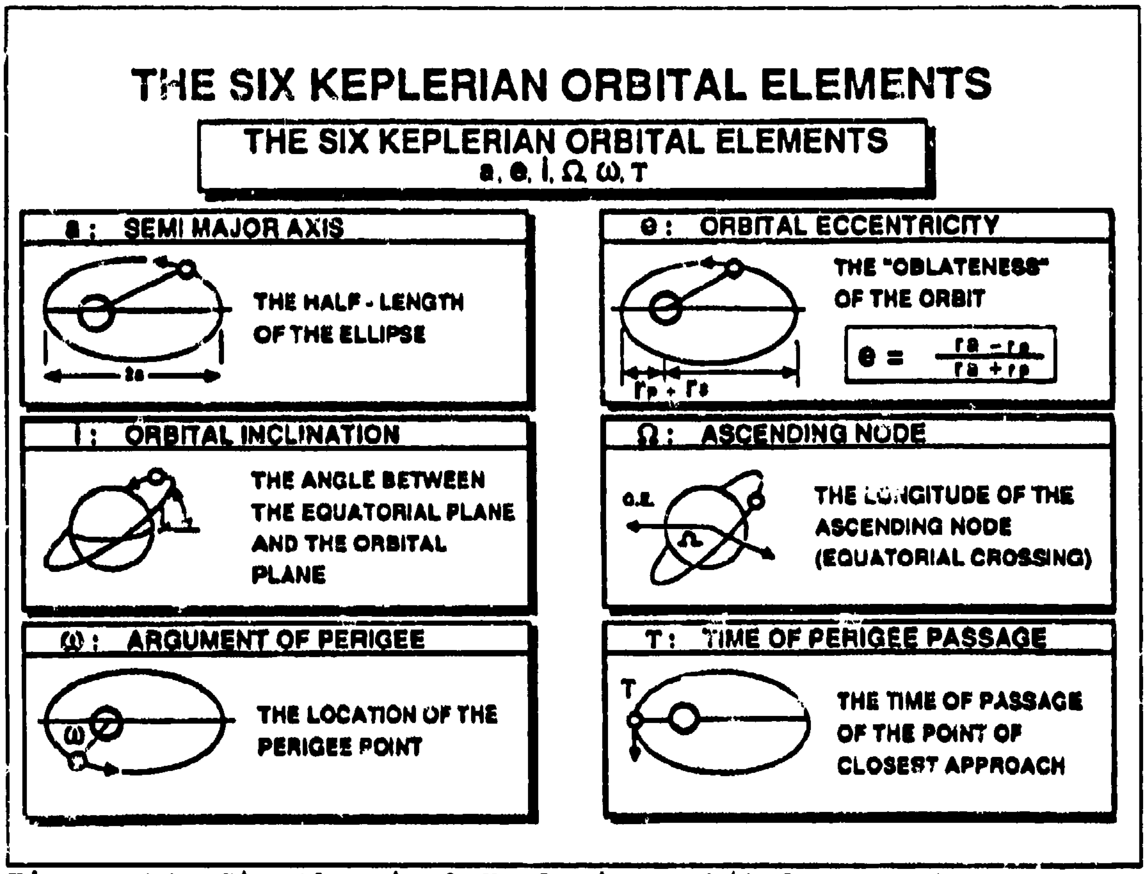

THE SIX KEPLERIAN ORBITAL ELEMENTS
THE SIX KEPLERIAN ORBITAL ELEMENTS
a, e, I, Ω, ω, T
a : SEMI MAJOR AXIS
THE HALF-LENGTH OF THE ELLIPSE
2a
e : ORBITAL ECCENTRICITY
THE "OBLATENESS" OF THE ORBIT
e = (ra - rp)/(ra + rp)
rp + ra
I : ORBITAL INCLINATION
THE ANGLE BETWEEN THE EQUATORIAL PLANE AND THE ORBITAL PLANE
Ω : ASCENDING NODE
THE LONGITUDE OF THE ASCENDING NODE (EQUATORIAL CROSSING)
ω : ARGUMENT OF PERIGEE
THE LOCATION OF THE PERIGEE POINT
T : TIME OF PERIGEE PASSAGE
THE TIME OF PASSAGE OF THE POINT OF CLOSEST APPROACH

Figure 26. Six Classical Keplarian Orbital Parameters. (Logsdon, 1992, p. 143)

are four unknowns and four simultaneous equations as shown in Equation (2):

$$
\begin{aligned}
P_\kappa^1 &= \sqrt{(X^1 - X_\kappa)^2 + (Y^1 - Y_\kappa)^2 + (Z^1 - Z_\kappa)^2} + c\,\Delta t_\kappa \\
P_\kappa^2 &= \sqrt{(X^2 - X_\kappa)^2 + (Y^2 - Y_\kappa)^2 + (Z^2 - Z_\kappa)^2} + c\,\Delta t_\kappa \\
P_\kappa^3 &= \sqrt{(X^3 - X_\kappa)^2 + (Y^3 - Y_\kappa)^2 + (Z^3 - Z_\kappa)^2} + c\,\Delta t_\kappa \\
P_\kappa^4 &= \sqrt{(X^4 - X_\kappa)^2 + (Y^4 - Y_\kappa)^2 + (Z^4 - Z_\kappa)^2} + c\,\Delta t_\kappa
\end{aligned}
\qquad (2)
$$

where $P_\kappa^\#$ is the pseudorange for the κth epoch in time, # is the satellite number, c is the speed of light and Δt_κ is the receiver clock bias from GPS time.

Parenthetically, it should be noted that although GPS time is synchronized within one microsecond of UTC, it does not recognize UTC leap seconds. The last simultaneous epoch of GPS and UTC time was in January 1980. Consequently, GPS time differs from UTC by the integer number of leap seconds since then.

E. GPS MESSAGE STRUCTURE

The entire GPS message format is illustrated in Figures 27 through 29. It is defined by ICD-GPS-200 written by Rockwell International who was the prime contractor for GPS. The basic message unit is called a Frame. It is subdivided into five subframes which are in turn subdivided into ten words each. A word has 30 bits. For positioning, a receiver requires only subframes one through three. Those contain the necessary ephemeris and clock corrections. However, the designers saw a need to provide the user information about all the satellites in the entire GPS constellation. They used subframes four and five to do this. Since this "Almanac" of extra information exceeded the size of those two subframes alone, it is transmitted using subframes four and five of 25 successive message frames. Each pair of subframes four and five of the 25 successive frames is referred to as a page. The entire block of 25 pages and subframes one through three is considered a Master Frame. Figures 27 and 28 illustrate this best. At a data rate of 50 bps, a frame needs 30 seconds and a master frame requires 12.5 minutes for complete transmission.

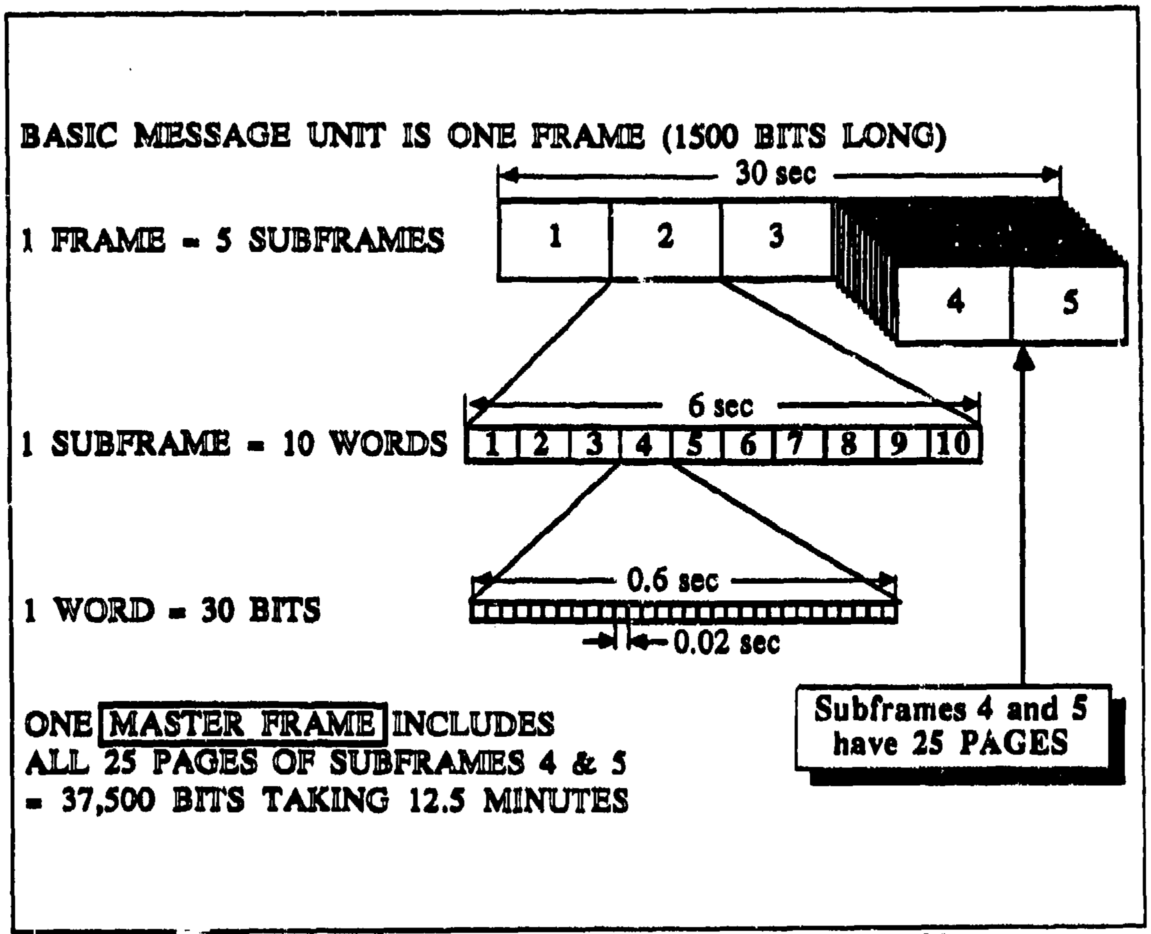

BASIC MESSAGE UNIT IS ONE FRAME (1500 BITS LONG)
30 sec
1 FRAME = 5 SUBFRAMES
1
2
3
4
5
6 sec
1 SUBFRAME = 10 WORDS
1 2 3 4 5 6 7 8 9 10
0.6 sec
1 WORD = 30 BITS
0.02 sec
ONE MASTER FRAME INCLUDES ALL 25 PAGES OF SUBFRAMES 4 & 5 = 37,500 BITS TAKING 12.5 MINUTES
Subframes 4 and 5 have 25 PAGES

Figure 27. GPS Message Format Breakdown. (Rockwell, 1987, p. 7.13)

F. GPS BIAS

There are numerous factors that affect GPS performance accuracy. Selective Availability is primary among these. While the exact nature of Selective Availability is classified, it is sufficient to know that it denies single receiver position accuracy of better than 100 meters 95% of the time. In addition, the Ionosphere and Troposphere affect the time of receipt by slowing down the signal. If the receiver could receive two frequencies, these atmospheric errors could be determined and eliminated. For the receivers that are capable of only single frequency reception (most receivers), compensation for these errors is achieved by using mathematical models which adequately reduce their affects for most positioning purposes.

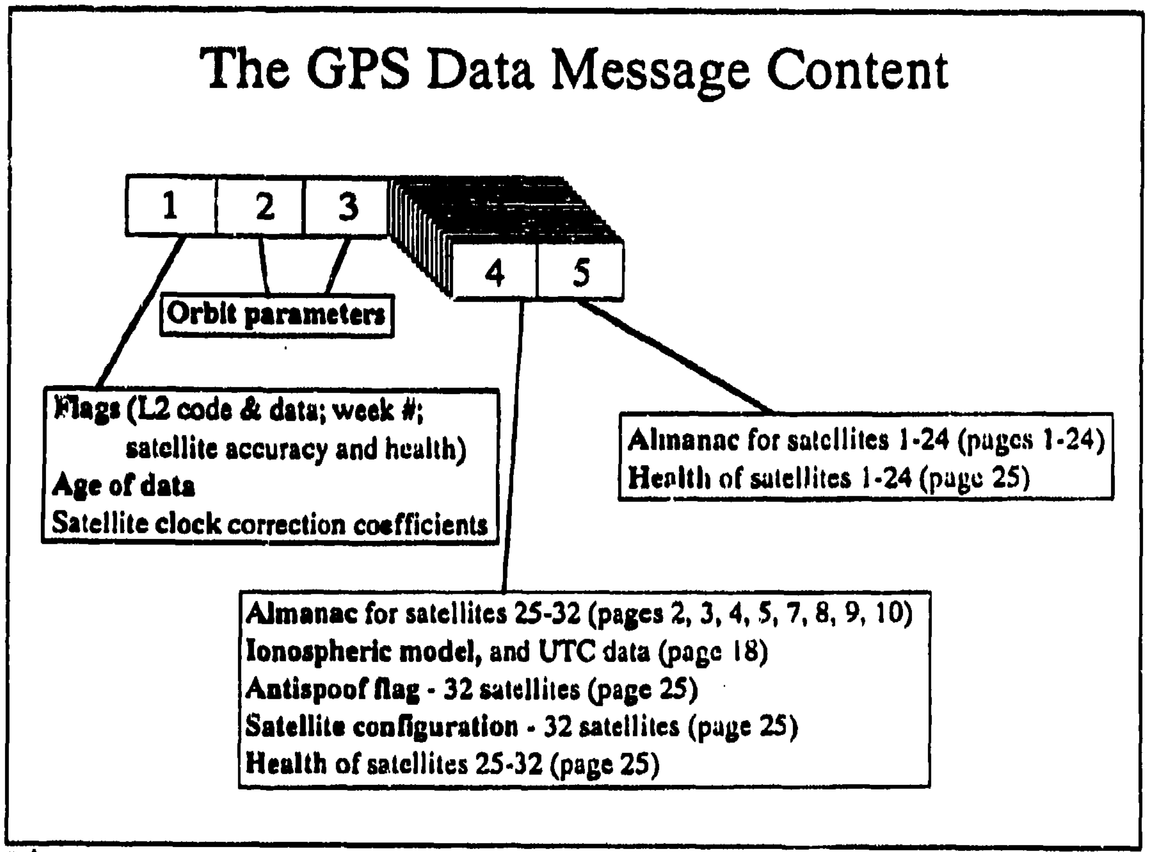

The GPS Data Message Content
1
2
3
4
5
Orbit parameters
Flags (L2 code & data; week #; satellite accuracy and health)
Age of data
Satellite clock correction coefficients
Almanac for satellites 1-24 (pages 1-24)
Health of satellites 1-24 (page 25)
Almanac for satellites 25-32 (pages 2, 3, 4, 5, 7, 8, 9, 10)
Ionospheric model, and UTC data (page 18)
Antispoof flag - 32 satellites (page 25)
Satellite configuration - 32 satellites (page 25)
Health of satellites 25-32 (page 25)

Figure 28. GPS Message Content Organization Within The Message Format. (Applied Research Laboratories, 1993, p. 2.2)

Signal reflections, commonly referred to as multipath, are another source of position errors. Multipath interferes with the fundamental concept of measuring the time from transmission to the time of receipt in order to determine range from the satellites. Fortunately, with adequate filtering and by paying careful attention in choosing the antenna, almost all the effects of multipath may be eliminated.

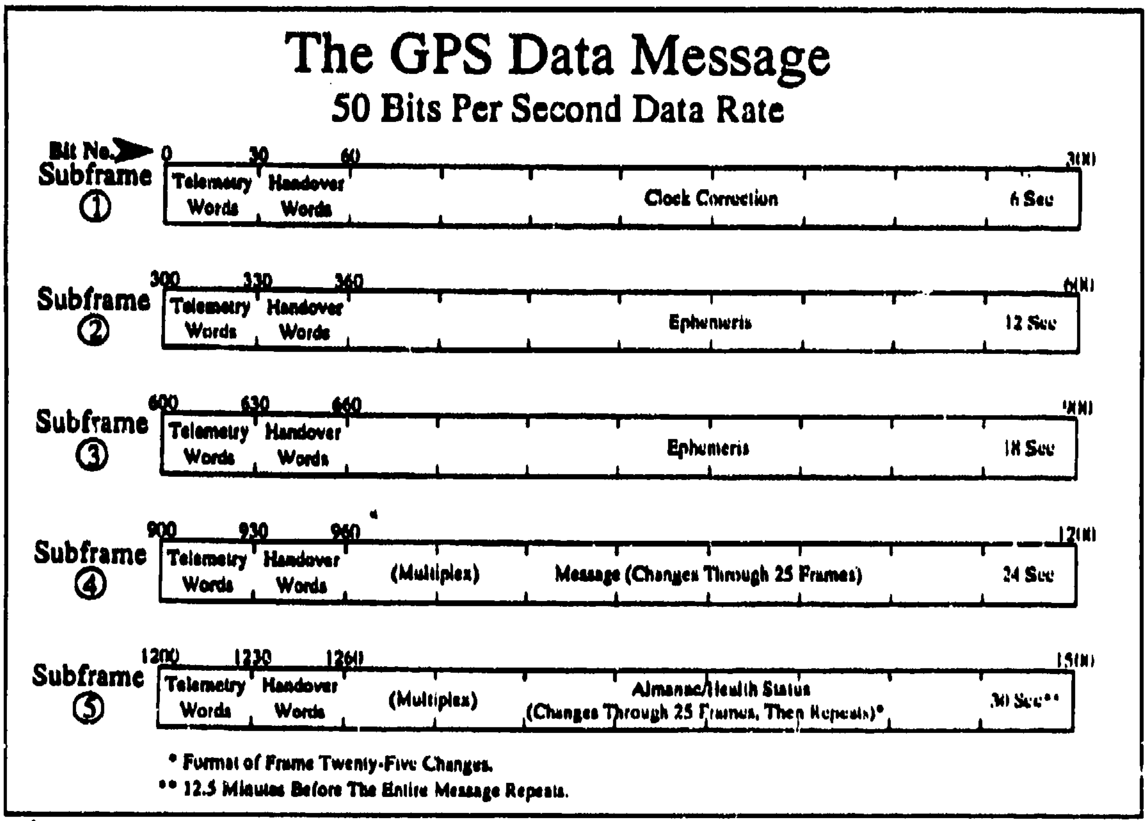


Figure 29 GPS Message Organization. (Applied Research Laboratories, 1993, p. 2.2)

Finally, satellite geometry relative to the receiver plays a role second only to Selective Availability in determining the precision achieved. The measure of "goodness" of satellite geometry is called Geometric Dilution Of Precision or GDOP. Optimal GDOP (the lower the GDOP number the better) is when three of the satellites are on the horizon equally spaced 120° apart and the fourth is directly over head as shown in Figure 30. Since the constellation is designed such that there are usually 12 satellites in view at all times, a low GDOP of two or three can regularly be achieved.

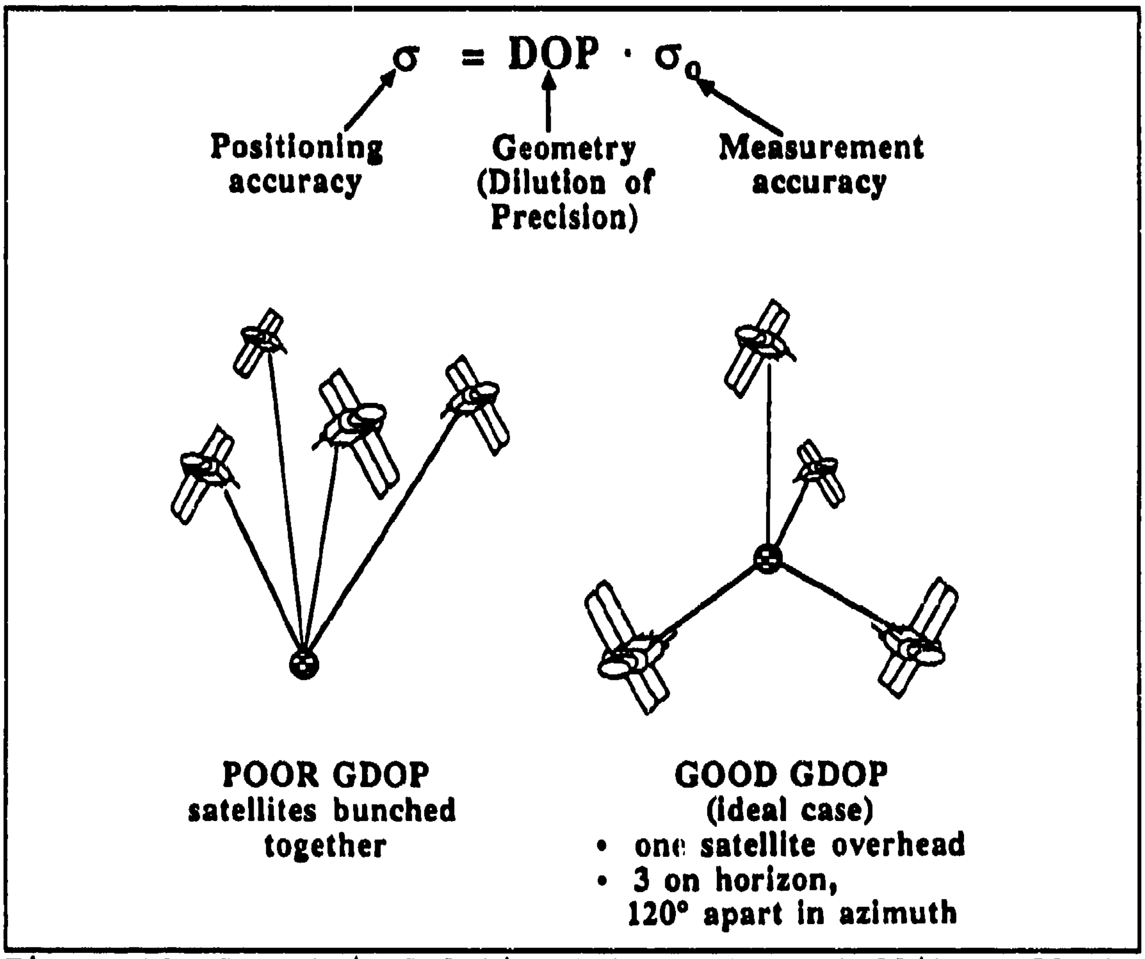


Figure 30. Geometric Relation Between GPS Satellites Affects The Position Accuracy. (Wells, 1987, p. 4.22)

G. DIFFERENTIAL GPS

SPS does not guarantee accuracies less than 300 m. It is possible to achieve accuracies two orders of magnitude better through a process called Differential GPS but it requires at least two receivers. The concept is simple and generates the greatest amount of promise in commercial applications of GPS. By placing one of the two receivers in a pre-surveyed location where its geodetic coordinates are known as precisely as possible, it can compare its true position to the position GPS provides and calculate the differences between latitude, longitude and altitude. If the resulting error signs are changed, the errors become corrections that may added to the second receiver's GPS position for a greatly improved position accuracy. Over baselines of up to 600 nm, the errors are approximately the same.

In practice, there are a few restrictions in employing DGPS. The most important is that the position results from both receivers must be derived using the same satellites or else the improvements in position are diminished by the introduction of new errors. The other restriction is that the corrections should be applied at the earliest point in the algorithm used to determine position coordinates in order to minimize the propagation of errors. The reasons for these restrictions will become clear after a brief description of the algorithm used to compute DGPS corrections.

The stationary GPS receiver is receiver is required to send corrections to the mobile receiver. It would be simple to subtract the true longitude, latitude and altitude from the geodetic position obtained from the GPS satellites. However, the errors introduced from SA, satellite geometry and atmospheric interference affect the psuedoranges from each satellite in varying degrees. One satellite psuedorange will always have a greater error than the next. Since the GPS position is calculated using a least squares reduction of the four simultaneous equations listed in equation two, all the individual psuedorange errors will be "blended" thereby permitting the greatest error to act on all the psuedoranges and the final position solution.

In order to avoid this problem, the stationary receiver computes the expected position of and range from each satellite (based on the time of travel of the individual signal) and compares it to the position and range that the GPS message asserts in its ephemeris. The difference between the two becomes a psuedorange correction. At least four of these corrections are transmitted to the mobile receiver and applied to the corresponding psuedoranges obtained by that receiver. Then the new psuedoranges are used in equation two to calculate the longitude, latitude and altitude of the mobile receiver. Since the individual corrections are applied before equation two, the final position is more accurate.

In practice, GPS makes all of its calculations using the Cartesian Coordinate ECEF system so the stationary receiver must translate its true position from geodetic coordinates to ECEF coordinates. In a similar manner, the mobile receiver must translate its corrected position from ECEF coordinates to geodetic coordinates. To summarize, the following steps are involved in improving the GPS position using DGPS:

1. Convert the true geodetic position of the stationary GPS receiver to ECEF coordinates.

2. Compute the satellite positions from the ephemeris.

3. Compute the expected satellite positions by multiplying the time from transmission to time of receipt by the speed of light.

4. Compare the received pseudoranges to the expected pseudoranges and calculate the pseudorange corrections for each satellite.

5. Transmit the pseudorange corrections to the mobile GPS receiver and add to the received pseudoranges.

6. Compute the mobile receiver's position using the corrected pseudoranges.

7. Convert the mobile receiver's position from ECEF coordinates to geodetic coordinates.

H. SUMMARY

The User segment of GPS draws together the results of several decades of study and experimentation. Hardware design and miniaturization are pushing the state of the art. Signal construction and frequency selection optimize its immunity to noise and interference. Orbital kinematics referenced to an ECEF coordinate system simplify and standardize the information contained in the GPS message structure. Even so, natural and man-made biases corrupt the integrity and accuracy of GPS. Through DGPS users can overcome the degradations caused by these biases and achieve real time accuracies unheard of heretofore. Understanding these principles upon which the User segment is founded will enable the user to better select a receiver that suits his purpose and integrate it into whatever application he so desires.

IV. GPS RECEIVER SELECTION AND INTERFACE WITH CPU

A. MISSION REQUIREMENTS FOR GPS

Before any serious search for a GPS receiver can be made, it is necessary to define the extent to which it will be used. These criteria come from the mission statement. This project's goals are to integrate an Inertial Measurement Unit (IMU) with GPS and other non-INS sensors through a smoothing and control filter to achieve autonomous flight of an Unmanned Aerial Vehicle using off the shelf hardware and technology. As a result, several requirements are defined for the GPS receivers.

To begin with, the position output should be frequent enough to provide regular corrections to the IMU. Also, the GPS receiver must actually update its position solution as often as possible to ensure accuracy in the information of its outputs. The receiver that is to be in the aircraft should be small and lightweight. It should be easily interfaced to the CPU, and it should provide the necessary information to conduct phase measurements of the incoming GPS signal. (This last feature should provide the positioning accuracy to auto-land the aircraft.) Finally, it must be readily available on the open market. Intrinsic in the mission statement is that it must be affordable given the project budget. With these guidelines, an informed selection can be made in choosing the best GPS receivers for the job.

B. GPS RECEIVER SELECTION

Having reviewed the principles of GPS operation and variations in receiver design features, an extensive survey of 216 different GPS receivers from 54 companies was conducted. In the January 1993 issue of *GPS World* magazine, many of the critical facets of each receiver were listed for comparison (Chan, 1993, pp. 52-65). In order to stay within budget, only those receivers less than $5,000.00 were considered. (The highest priced receiver was $55,000.00 with many between $10,000.00 to $20,000.00.) Among the remaining units, those that were capable of tracking six or more satellites simultaneously were selected for comparison. This left the list at seventeen receivers.

Of the remaining seventeen receivers, all were capable of the same approximate accuracy and information output rate at once per second. The one notable exception was Navstar's XR5 with an output rate at 1/10 second, but ultimately, it was not chosen because, at $1000.00, it cost three times as much as the Motorola PVT-6. Since all the receiver units performed at virtually the same level, the deciding factors came down to cost, weight, size and power consumption. Table I shows the top eight receivers with these features listed.

Table I. COMPARISON OF TOP EIGHT CANDIDATES FOR THIS PROJECT'S GPS RECEIVERS. (CHAN, 1993, PP. 52-65)

GPS Maker	*RCVR NAME*	*Max # chan*	*Weight ·)*	*Size (in^3)*	*Power (Watts)*	*Price*
Furuno	GB-92/ GN-72	8	2.8	8.74	2	$750
Magellan	GPS Brain	5	4.8	6.53	0.9	$445
Magnavox	Turbo Eng	6	3.2	11.44	<1.5	$3750
Motorola	PVT-6	6	4.5	4.64	1.3	$369
Navstar	XR5 Kernel	12	5	8.28	2.5	$995
Novatel	Perform OEM	10	6.2	15.44	5	$3195
Sony	IPS-2010	8	8	15.65	2.61	$1000
Trimble	SV6	6	2.9	2.85	1.85	$995

The Trimble price quoted includes a starter evaluation kit. The Motorola receiver, when all options were installed, was actually more expensive. By the criteria used, Trimble had the better unit would have been chosen over Motorola but they were very slow in answering inquiries for characteristics and pricing information. When the decision was made, they were not considered. However, two factors make the choice of Motorola's PVT-6 the best selection. One, they also make a portable differential GPS base station called a LGT-1000. It is conceivable, in the future, this project will choose to experiment with the specialized equipment. And two, Trimble has since stated they do not intend to pursue the small, single receiver user market. Instead, they will specialize in industrial scale receivers for use at airports and in surveying.

C. GPS RECEIVER DESCRIPTION

1. Hardware

The Motorola PVT-6 is an original equipment for manufacturer (OEM) module. Although evaluation software is provided in a starter kit, it is not useful when the CPU is multi-tasking with several other processes. PVT-6 stands for Position, Velocity and Time with six channels. Figure 31 shows the dimensions of the unit without the antenna. The specifications are listed in Appendix C. The PVT-6 is capable of tracking six satellites simultaneously. It uses only the C/A coded message from the L1 carrier. The code tracking is carrier aided. It can be powered by a 12 Vdc unregulated or 5 Vdc regulated source. (Motorola, 1992, p. 3-2)

The PVT-6 is supplied with a low profile antenna of a microstrip design called a strip patch antenna. The antenna

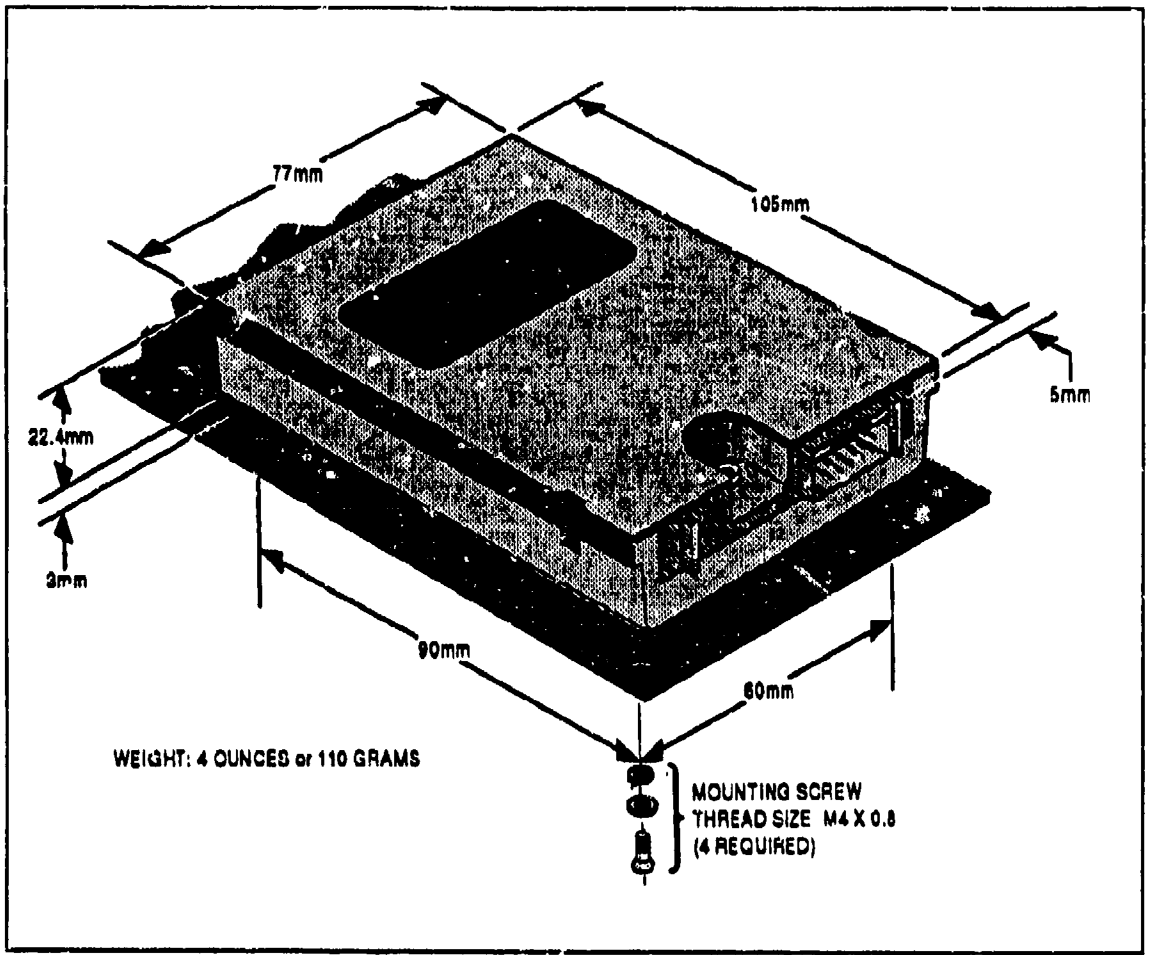


Figure 31. Physical Dimensions Of The PVT-6.(Motorola, 1992, p. 3-3)

gain pattern is provided in Appendix D. The L1 frequency is collected by the antenna, passed through a narrow band filter, amplified and sent to the receiver unit via an RG-58 coaxial cable. It provides the antenna 5 Vdc @ 25 mA from the receiver and returns the amplified L1 signal from the antenna. It should be noted the power loss along the cable should not exceed 6 dB at a frequency of 1575.42 MHz which equates to a maximum length of six meters for RG-58 coaxial cable. RG-58 is the recommended cable because it allows a sufficient length of cable while achieving lighter weight due to the small gauge without the corresponding losses due to increased resistance that accompany even smaller gauge coaxial cables at that frequency. Antenna placement should position the patch plane to be as level as possible and have an unobstructed view of the horizon. This ensures a direct line of sight to all the visible satellites. Table II and Figure 32 provide the power and data connector pin assignments. (Motorola, 1992, p. D-1)

Table II SERIAL PORT AND POWER PIN ASSIGNMENTS. (MOTOROLA, 1992, p. 3-2)

Pin #	Signal Name	Description
1	+12V/+5V BATT	(Optional) +5 Vdc regulated or +12 Vdc unregulated for running Real Time Clock and retention of satellite ephemeris information stored in Static RAM memory.
2	+5V MAIN	(Optional) +5 Vdc regulated for power requirements of the entire GPS receiver(+12 Vdc signal not used).
3	+12V/+5V RTN	Power supply (+5V or +12V) return.
4	V_{pp}[1]	Flash memory (EPROM) programming voltage.
5	+12V MAIN	+12Vdc unregulated for power requirements of the entire GPS receiver.
6	1 PPS[2]	(Option A) 1 pulse per second output.
7	1PPS RTN	(Option A) 1 pulse per second return.
8	RS232 TXD	Serial RS232 data output.
9	RS232 RXD	Serial RS232 data input.
10	RS232 RTN	Signal return for RS232 signals.

[1]Not used (used only by Motorola for updating of software).
[2]1 Hz pulse train with rising edge coincident with UTC/GPS 1 second tick.

As Figure 33 shows, the L1 analog signal enters the receiver and is down converted to an intermediate frequency

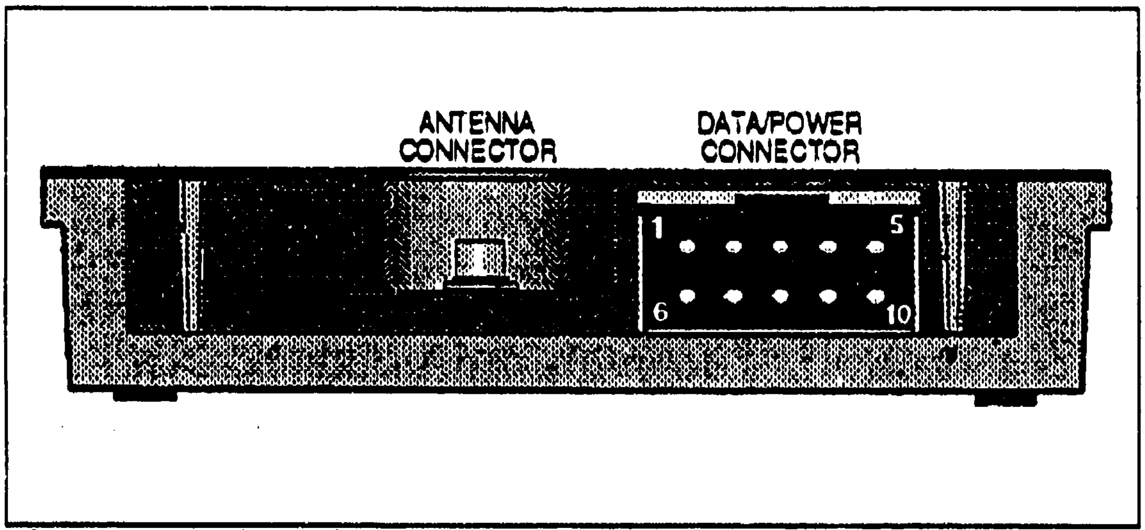


Figure 32. Side View Of The PVT-6 Showing The Pin Numbering Order. (Motorola, 1992, p. 3-2)

(IF). The IF is passed to the 6-channel code and carrier correlator section where it is digitized and split for input into six separate channels for correlation, filtering, carrier tracking, code tracking and signal detection. The output is synchronously routed to the MPU (PVT-6 CPU) where the information is processed and formatted. Also within the MPU is the RS-232 interface for full-duplex, asynchronous data communication with the host equipment. (Motorola, 1992, p. 2-2)

The PVT-6 is configured to operate at 9600 baud with eight bit words, one start and stop bit and no parity. Its communications port is considered Data Communications Equipment with respect to the RS-232 serial communications standard. However, close examination of the available pins for serial communications reveals the presence of only the TX (pin 8), Rx (pin 9) and Gnd (pin 10) pins. This dictates there

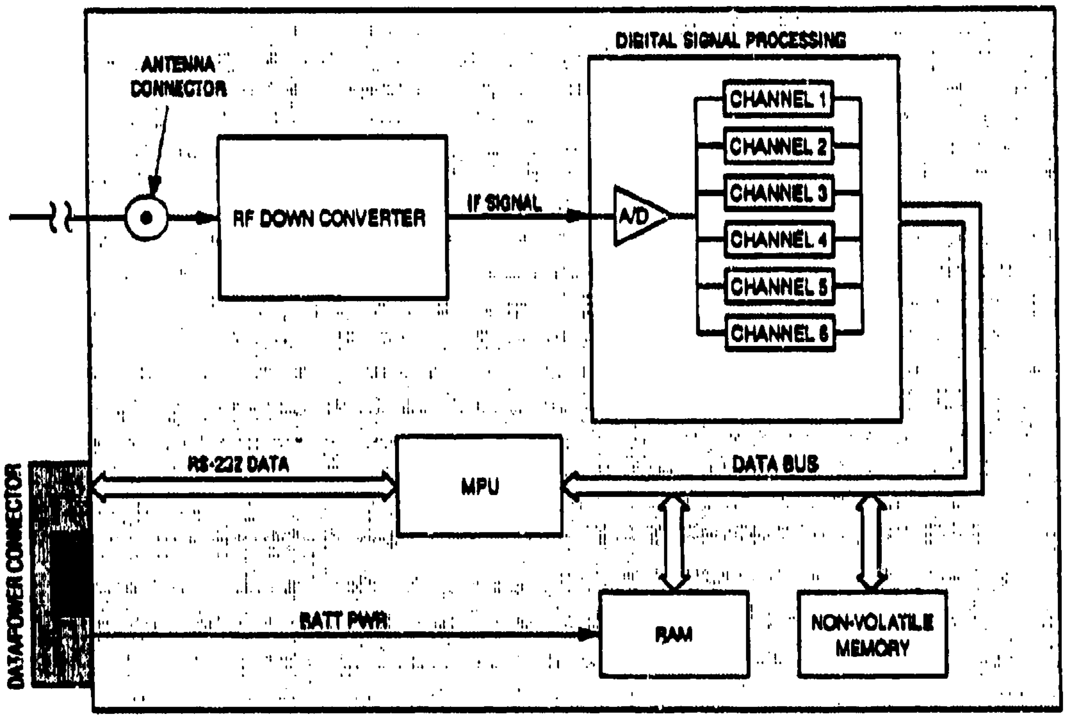


Figure 33. Functional Block Diagram Of The PVT-6. (Motorola, 1992, p. 2-1)

will be no modem line configuration (DTR and DSR) and at least no hardware flow control (CTS and RTS). In fact, there is no software flow control (XON and XOFF) either implying the Data Terminal Equipment (DTE- host equipment) must monitor the port continuously or miss valuable information. Likewise, the PVT-6 must do the same. In practice, it receives all serial input into a two kilobyte buffer from which it reads 750 bytes per second. The buffer is large enough that it never overflows. (Motorola, 1992, pp. 3-2 - 4-3)

Finally, the PVT-6 is provided with static random access memory (SRAM) for retention of the satellite ephemeris data. To prevent the loss of this information, an external power source such as a battery may be hooked up to the +12V/+5V BATT connection (pin 1) when the GPS unit is powered off. In addition, nonvolatile EEPROM is used for storage of custom operating parameters, almanac information and other information such as the receiver serial number and option list. (Motorola, 1992, p. 2-2)

2. Motorola Binary Message Format

The PVT-6 receiver is capable of providing its information in any one of three formats: Loran, NMEA-0183 and Motorola Proprietary Binary (Motorola, 1992, pp. 4-2 - 4-6). Both Loran and NMEA (National Marine Electronics Association) would severely restrict this project's flexibility and accuracy, and so are not used. Motorola Binary format is logically organized and provides tremendous flexibility to the user to apply GPS to what ever purpose.

Motorola's Binary format has 47 different messages it sends and receives ranging in length from 7 to 294 bytes. Each message begins with "@@" (double ASCII 0x40) which is followed by a two letter, case sensitive message ID. Then, 0 to 287 bytes of data follow, a byte size check sum and an ASCII carriage return-line feed (0x0D 0x0A). The check sum is the exclusive-or of all the preceding bytes in the message. The data that follows the message ID can be thought of as scaled floating point data stored in integer format. This avoids the floating point storage convention as well as saves space (only one byte is needed instead of four or more). The user must be very careful to observe the units of measurement provided in the Motorola GPS Technical Reference Manual. (Motorola, 1992, p. 4-2)

For every command to the GPS receiver, there is a response usually using the same message ID. The receiver has no way to determine if the information it has received from the CPU is correct other than by verifying the several intrinsic properties of the message itself:

- The message begins with "@@".
- The message ends with CR-LF.
- The length of the message matches that expected for the message ID.
- The checksum is valid.
- The message data is within the expected maximum and minimum ranges.

If the command input to the PVT-6 changing a parameter is invalid for any of the above reasons, it responds as expected, but the data values will not reflect the ordered change. It is prudent, therefore, to compare every input message with the subsequent PVT-6 response. (Motorola, 1992, pp. 4-2 - 4-3)

Input commands may be of the type that change a particular configuration parameter of the receiver. Any command with a message ID that begins with a upper case "A" falls into this category. Input commands may also be of the type that enable or disable the output of data or status messages. They are the commands that the CPU will use for measuring position, velocity, time, pseudorange and satellite ephemeris data. These output status messages include 12 that are the work horses of the PVT-6. Table III shows only nine because three of the commands are the reciprocals used by the CPU to talk to the PVT-6. The nine shown are listed in order of precedence for transmission from the GPS receiver to the CPU. (Motorola, 1992, p. 4-3)

Table III DATA MESSAGE OUTPUT RATES SHOWN IN ORDER OF PRECEDENCE OF TRANSMISSION FROM THE PVT-6 TO THE CPU. (MOTOROLA, 1992, P. 4-4)

OUTPUT MESSAGE TYPE	CONTINUOUS ([illegible])	ONE TIME ([illegible])
Position/Channel Status	At selected update rate	When requested
Satellite Range Data Output	At selected update rate	When requested
Pseudorange Correction Output	At selected update rate	When requested
Ephemeris Data Output	When Eph data changes	When requested
Visible Satellite Status	When Vis data changes	When requested
DOP Table Status	When DOP data changes	When requested
Almanac Status	When Alm status changes	When requested
Almanac Data Output	When Alm data changes	When requested
Leap Second Pending		When Requested

3. PVT-6 Operation

The PVT-6 has some limitations that, if understood, may be overcome. Data is output once per second from the receiver. This information is calculated from the ephemeris and time measurements of the previous second time tic called the measurement epoch. In order to provide the most accurate position given the latency of data, the position, velocity and time represent the best estimate of the data advanced one second. Figure 34 graphically illustrates this process. The first byte of information for a particular measurement epoch is transmitted anywhere from zero to 50 ms after the subsequent measurement epoch. Only the time for the Satellite Range Data message and the Pseudorange Correction message is not propagated forward. Careful consideration addressed to this latency issue in the design of any Kalman Filter or smoothing filter that integrates GPS position with the IMU and other sensors will improve performance. (Motorola, 1992, pp. 4-13 - 4-17)

To assist in synchronization of the GPS data with other sensor data, a one pulse per second (1PPS) signal is output using pins six and seven of the serial port. The pulse is approximately 200 milliseconds long with its leading edge corresponding precisely to the measurement epoch. The PVT-6 has the option of additionally offsetting the 1PPS up to 0.999999999 seconds. If there is a requirement for the 1PPS to be precisely synchronized with the UTC or GPS measurement epoch, the user may also offset the receiver measurement epoch as well as account for antenna cable delay. For this project, these latter conveniences will not be required. This is shown

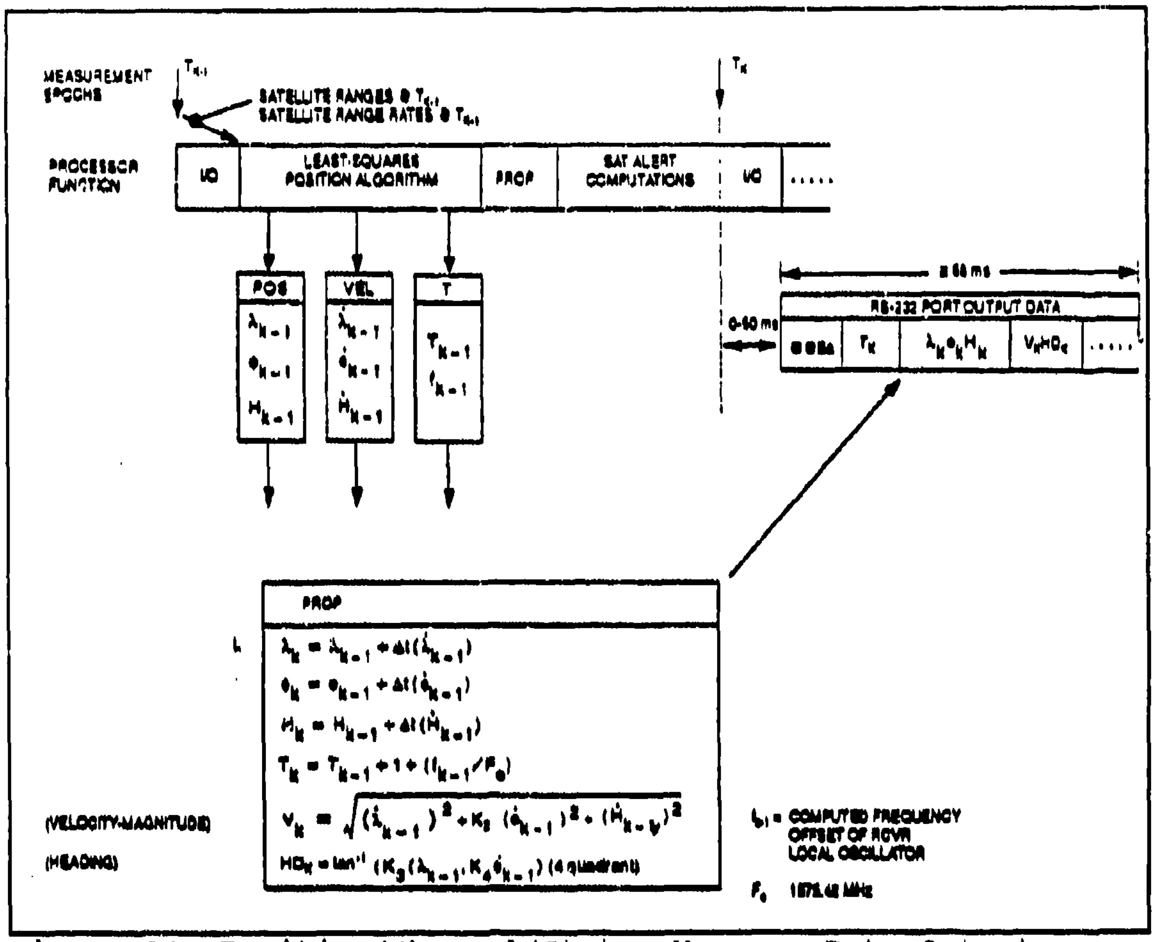


Figure 34. Position/Channel/Status Message Data Output Latency. (Motorola, 1992, p.4-15)

in the timing diagram in Figure 35. (Motorola, 1992, pp. 4-16 - 4-17)

Finally, the PVT-6 suffers from periodic outages like most other receivers regardless of quality. These outages may be caused by temporary loss of satellite track do to obscuring by a building, tree or wing tip if the aircraft were in a banking turn. Sudden accelerations may also cause loss of track. In this event, the time for the receiver to reacquire the track and output data is important. As shown in Table IV,

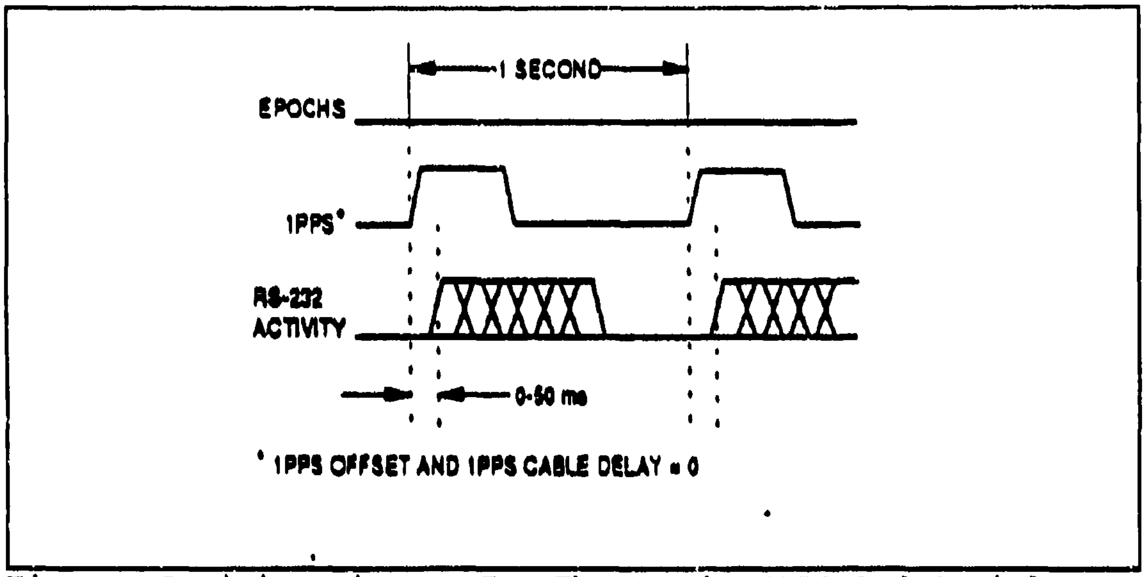


Figure 35 Timing Diagram For The Epoch, 1PPS And Serial Output Data. (Motorola, 1992, p. 4-17)

this time may take as long as 15 seconds. If the aircraft were attempting to land using GPS alone, this could be fatal. (Motorola, 1992, p. 4-7)

Table IV NOMINAL REACQUISITION TIMES FOR THE PVT-6. (MOTOROLA, 1992, P. 4-7)

TIME OBSCURED	REACQUISITION TIME
15 sec	< 12 sec
30 sec	< 13 sec
45 sec	< 14 sec
60 sec	<15 sec

D. INTERFACING THE GPS RECEIVERS

The two GPS receivers to be used for DGPS will be linked to the ground CPU and the UAV CPU through RS-232 serial ports. The big picture of the project showing where GPS fits in is shown in Figure 36. The ground control station will be interfaced through a 33 MHz Intel 80486DX personal computer. It will be the primary data flow conduit for the uplink and downlink as well as provide formatting and mass storage. The additional link in the ground station shown in Figure 36 is for redundancy in the event the main link fails and emergency control of the UAV is required.

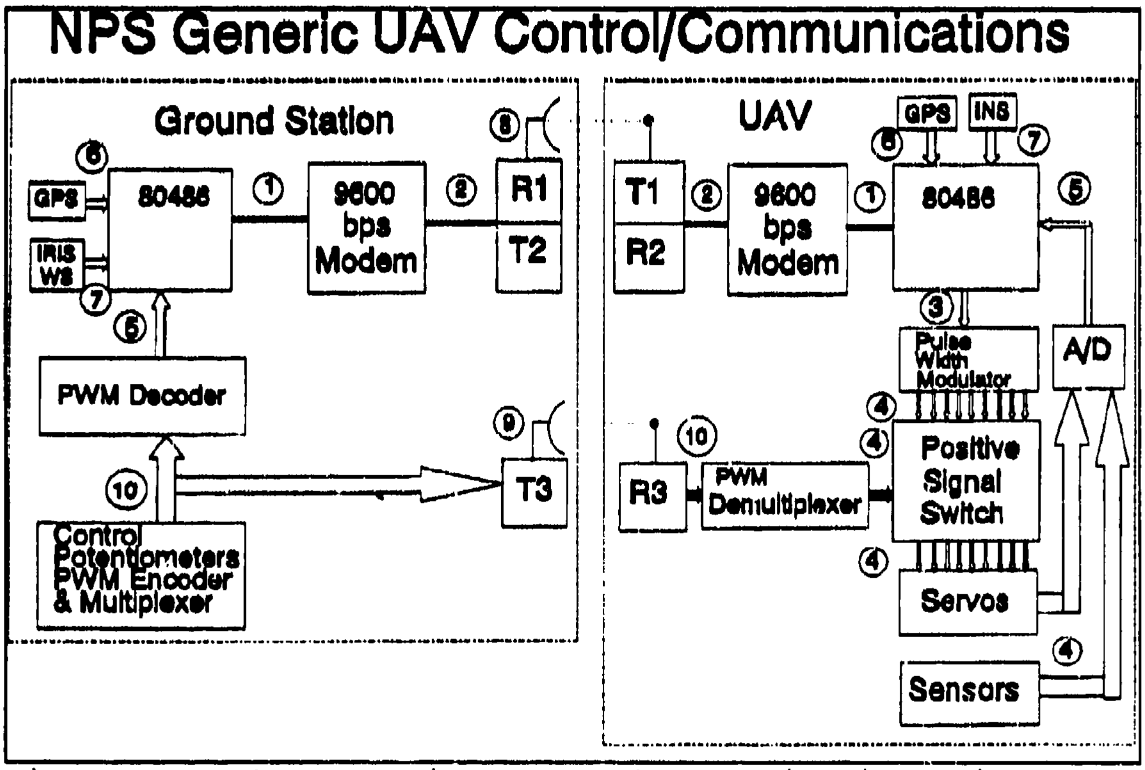


Figure 36 Archytas Project Control/Communications Diagram Showing Where GPS Fits Into The Big Picture.

The UAV CPU uses off the shelf hardware that economizes on size, weight and price. This is done by using six computer cards and a passive backplane. The CPU is a 33 MHz Intel 80486DX with 256 Kb double cache memory. Non-volatile memory for the operating system and programs is provided by a 2.88 Mb RAM/ROM Disk card instead of a relatively heavy hard drive. In addition, servo control and sensing are managed using an Analog-to-Digital (A/D)/Digital-to-Analog (D/A) card and a ten channel Timer/Counter card. Serial I/O is handled through the two standard serial communication ports on the CPU card (COM1 and COM2) and an intelligent 8 port RS-232 serial I/O card. The GPS receiver communicates to the CPU through port 1 on this card (not to be confused with COM1 on the CPU card). In order save space, all the cards are connected to a common bus using a passive backplane. A simplified drawing of the final arrangement is shown in Figure 37.

1. PCL-744 8-Port Intelligent RS-232 Interface Card

The UAV CPU needs multiple serial ports for other data inputs from the link modem, the IMU and the non-INS sensors. These multiple serial inputs are provided through Advantech's PCL-744 8-Port Intelligent RS-232 Interface Card.

The PCL-744 is an intelligent card in that it has its own processor and 64 Kb RAM buffer. It requires only one interrupt number fc: eight ports although each may configured independently. This configuration includes determining the

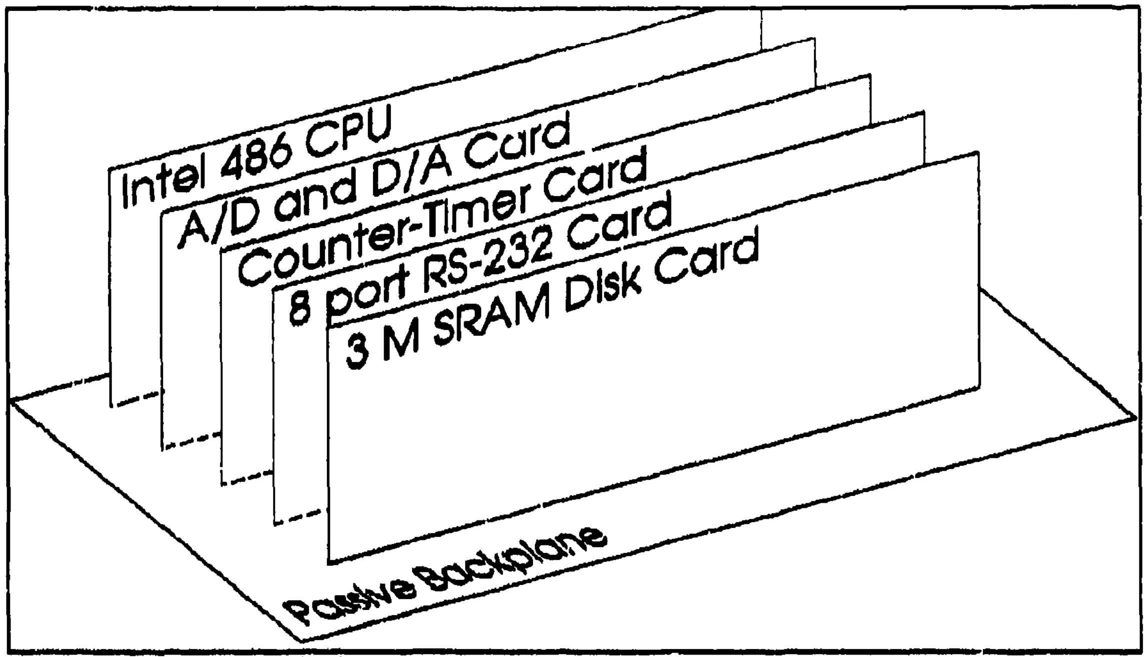


Figure 37 Simplified Illustration Of UAV CPU Hardware.

size of the transmit and receive ring buffers and what type of flow control is used. These facts enable the UAV CPU to perform multi-tasking by allowing it to access the ports and process data without losing any information from the other ports arriving simultaneously. The PCL-744 takes care of that by multiplexing between its eight ports and placing the incoming data in buffers that can be accessed by the UAV CPU.

The PCL-744 comes with a library of 40 user functions written in several high level languages including Microsoft C and Turbo C by Borland. This project uses the Turbo C routines for all serial communications. These functions allow the user to use the card in anyway desired for serial I/O. It is also necessary to ensure the TSR drivers are loaded prior to PCL-744 operation. They are called STD-DRV.EXE and 744-DRV.EXE and should reside in the AUTOEXEC.BAT file to ensure their loading.

The description of the software drivers for the GPS receivers that follows is the same in functional design for both the ground and UAV units, but there are slight differences in coding. This is due to the fact that the UAV receiver must communicate through the PCL-744, and the ground receiver communicates directly with the CPU through a standard DOS COM port. The port interface for the ground CPU uses a standard C library interrupt routine, and the port interface for the UAV that communicates through the PCL-744 uses the PC-ComLIB serial communication library routines provided with the card.

2. GPS Software Driver

In communicating with the PVT-6, all transmissions are without any flow control. It is necessary, therefore, to provide a storage buffer where the contents of the transmission may be stored for holding until the CPU is ready to handle the data. The information sent to the PVT-6 is put into a two kilobyte buffer in the receiver itself. All message handling is done by the receiver including parsing, verification and interpretation, and is transparent to the user. The user's only responsibility is to ensure that the message sent to the receiver is properly formatted. However, when receiving messages from the PVT-6, parsing, verification and interpretation must be done by the user. This allows flexibility for the user and is one of the minor reasons for selecting a Motorola GPS receiver.

This section describes the software driver required to accept information from the GPS receivers. The software in Appendix E is written in modular format to allow flexibility and ease of handling. It is structured so it can be expanded and built upon. The following paragraphs describe the flow chart shown in Figure 38 which illustrate the concept of the GPS software driver in Appendix E and the structure of the software driver.

The PCL-744 requires the eight ports to be initially configured using the vendors software. This configuration is stored in the PCL-744 driver and the card is thus reconfigured upon startup every time power is applied. This does not make the card ready for operation, though. Each port must be individually opened and set to the communication parameters required by its user. For instance, the PVT-6 uses 9600 BAUD with an eight bit word, no parity and one start/stop bit. In addition, RS-232 modem control (i.e., DTR and RTS), hardware control (i.e., DTS and RTS) and software flow control (i.e., XON/XOFF) are set to off. This is all accomplished using the function "Open_ports". At the conclusion of using the ports they should be formally closed and this is accomplished by using the function "Close_ports".

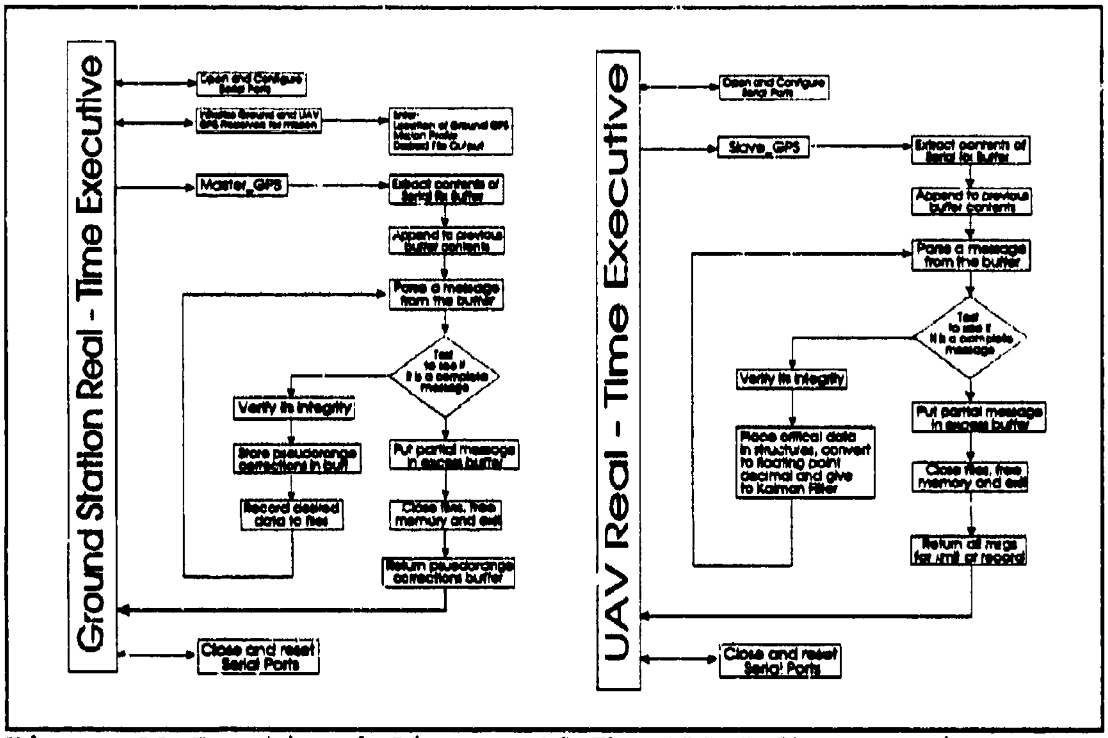

Ground Station Real - Time Executive
Open and Configure Serial Ports
Master_GPS
Extract contents of Serial Rx buffer
Append to previous buffer contents
Parse a message from the buffer
Test to see if it is a complete message
Verify its integrity
Store pseudorange corrections in buff
Record desired data to files
Put partial message in excess buffer
Close files, free memory and exit
Return pseudorange corrections buffer
Close and reset Serial Ports
UAV Real - Time Executive
Open and Configure Serial Ports
Slave_GPS
Extract contents of Serial Rx buffer
Append to previous buffer contents
Parse a message from the buffer
Test to see if it is a complete message
Verify its integrity
Place critical data in structures, convert to floating point decimal and give to Kalman Filter
Put partial message in excess buffer
Close files, free memory and exit
Return all msgs for xmit or record
Close and reset Serial Ports

Figure 38 Functional Diagram Of The GPS Software Drivers For Receiving Information From The PVT-6.

The main functions that handle incoming information are called "Master_gps" for the ground CPU and "Slave_gps" for the UAV CPU. The "Master_gps" function first gets the buffer length of the message waiting in the Rx buffer for port number one on the PCL-744. It then creates a buffer of exactly that size and extracts the information from the Rx buffer and puts it into the temporary buffer just created. Next, it creates the input buffer which will contain the temporary buffer appended to the excess buffer contents. The excess buffer has the partial message left over from the previous call to "Master_gps". With the input buffer created and the contents of the excess and temporary buffers copied into it,the excess and temporary buffer memory is freed.

The pointer to the input buffer is then passed to the function "Parse_inbuff" where the first message is parsed from the input buffer and returned. If a complete message is not returned, the function places the partial message in the static excess buffer, frees the memory for the input buffer and exits the function "Master_gps". If a complete message is returned, it is passed to the function "Msg_verification" where it is checked for the correct length, check sum, ASCII prefix "@@" and hexadecimal suffix x0D x0A (ASCII carriage return and line feed). When it has been verified, the message is placed in a buffer to be passed out to the calling routine for transmission to the UAV GPS if it is a pseudorange correction. Otherwise, it is written to disk for post-processing.

In the UAV GPS driver, the "Slave_gps" function operates exactly the same way except after a message has been verified, it is put into a structure and scaled according to the format in the Motorola protocol for subsequent use by the Kalman filter. One of the more complicated structures for the ephemeris message has been written as an example by Motorola and incorporated in the header file GPSCONV.H. All message output from the UAV GPS is passed out to the calling routine to be down linked to the ground in binary format (to reduce downlink time) for post-processing.

Two other functions are mentioned but not yet completely written which play an important role. They are the "Initialize_master_gps" and "Initialize_slave_gps" functions. They will pre-configure the receivers for such things as true geodetic coordinates of the ground antenna, time, date, output rates, desired files to be collected etc. They should be made interactive to allow the user to change and set any parameters. The code framework for these functions is in the file GPSPORTS.H in Appendix E.

Finally, the source file GPSAN.C with GPSCRNCH.H was used to post-process data collected for this thesis. It was written to take the Position/Channel/Status message from an ASCII file, convert it to binary and then place the message in a structure where the scaled and formatted data could be analyzed. Then GPSCRNCH.H "crunched" the numbers and computed the maximum, mean and standard deviation for the various time delays.

V. VERIFICATION OF OPERATIONAL CAPABILITIES

GPS is a complicated new tool for navigators. Each receiver manufacturer employs unique and innovative techniques to improve its accuracy and flexibility. Motorola provides specifications and operational capabilities for its receiver and users have published various accuracies achieved using DGPS. It is prudent to verify those features important to this project's success. Those two capabilities are examined below. They are:

1. Verification of the manufacturer's reacquisition time specifications.

2. Determination of the effects of pseudorange correction delay in getting from the stationary receiver to the mobile receiver.

In the following sections, these concerns are investigated. For each case, the experimental setup is described and the results are presented.

A. REACQUISITION TIME VERIFICATION

This project intends to use DGPS with one of the two antennae on a UAV. During some flight maneuvers, it is expected that the GPS antenna will be masked from one or more of the satellites it is using to calculate its position. It is important for the designers of the Kalman Filter to understand the normal reacquisition times given such a circumstance. In

Table IV in Chapter IV, Motorola presents its results for the reacquisition time of the PVT-6 receiver given the antenna was masked for 15, 30, 45 and 60 seconds. The method they used to collect this data is not well documented, and it is not indicated whether or not the receiver was in three dimensional fix mode or two dimensional fix mode. The difference could be significant because the former requires four satellites for a fix and the latter only three.

The reacquisition time was investigated using this project's PVT-6 receiver. Table V shows the results next to the Motorola claims. Data was collected using the software supplied by Motorola with an evaluation kit. The antenna was completely masked 360° around above and below using four to six layers of tin foil. The mask was left in place for the specified amount of time and then removed for at least 30 seconds. Data files were imported into a spread sheet where the time marked data was analyzed.

The results do not exactly agree with those that Motorola publishes, but they are close. Significantly, data obtained during daylight hours were within one standard deviation of Motorola's claimed reacquisition times whereas data collected after daylight hours were not. No effort was made to confirm this suggested phenomenon nor was any effort made to correlate it to ionospheric thickness or tropospheric conditions such as rain or cloudiness. However, until the IMU is integrated with GPS using a Kalman Filter and the effects of a two to three second delay in reacquisition time can be determined, investigation of this discovery is left for further study.

Table V COMPARISON OF MOTOROLA AND EMPIRICAL REACQUISITION TIMES.

Time Obscured (sec)	*Motorola Reacquisition Time (sec)*	*Empirical Reacquisition Time (sec)*	*Standard Deviation (sec)*
15	<12	13.75*	2.2
30	<13	17.08**	2.0
45	<14	16.67**	3.2
60	<15	13.67*	2.1

* *These observations were taken from 1230 to 1500.*
** *These observations were taken from 2200 to 2400.*

B. DGPS ACCURACY AND PSUEDORANGE CORRECTION LATENCY

The accuracy that DGPS can achieve is well documented. The PVT-6 is designed and built to permit DGPS. In fact, the Motorola Binary format includes master/slave DGPS commands and automatically incorporates the data in its calculations. It is worthwhile to validate the expected results especially in light of the fact the communications software to be used is not written by the manufacturers and may adversely affect the accuracy.

In designing the test for DGPS, the length of the baseline between the two receivers was an issue that was researched. The results indicated the baseline range between the two receivers did not significantly affect DGPS accuracy up to about 50 km. Beyond 50 km, out to about 1000 km, the effects slowly increased the error (Kremer, 1990, pp. 307-312). This project's maximum baseline is not expected to exceed 15 km. For all intents and purposes, a zero meter base line is as good as a 1000 meter baseline and certainly is easier to test. Therefore, this test simply mounted the two antennae side by side at a pre-surveyed locatior (See Appendix F). Data was collected by sending pseudorange corrections from the master antenna to the slave antenna using the software driver in Appendix E. The PVT-6 puts out a position message at a rate of once per second. These positions were differenced using the ECEF equations in Appendix A, and the mean, maximum and standard deviation were calculated.

Table VI shows the DGPS accuracy with various delays in transmitting the correction to the slave receiver and Figure 39 graphically illustrates the same results. The delay was artificially introduced through the function call to "Pr_delay" in the header file GPSDELAY.H. The delays were in integer amounts of seconds as shown below.

Interestingly, Motorola technicians quote adverse effects of pseudorange corrections received after 90 seconds. The results above suggest an immediate degrading effect of any amount of delay and that the error grows with delay time. It is hypothesized that the decrease in error at the 15 second delay point is a spurious result caused by any number of factors. It is left for further study to confirm this.

Table VI OBSERVATION OF PSEUDORANGE CORRECTION INPUT DELAY ON THE EFFECTIVE ACCURACY OF DIFFERENTIAL GPS.

Delay (sec)	*Average Offset (m)*	*Maximum Offset (m)*	*Standard Deviation (m)*
0	1.17	3.76	1.00
1	2.58	5.16	0.87
2	2.75	3.84	1.00
3	2.72	3.86	0.96
4	0.96	3.28	0.92
5	2.41	5.53	0.97
6	1.85	3.85	1.39
7	2.45	8.03	2.02
8	4.07	8.83	2.14
9	4.51	10.80	2.19
10	4.94	13.85	3.36
15	1.96	5.91	1.48
30	6.09	21.21	4.59
45	21.40	42.16	10.90

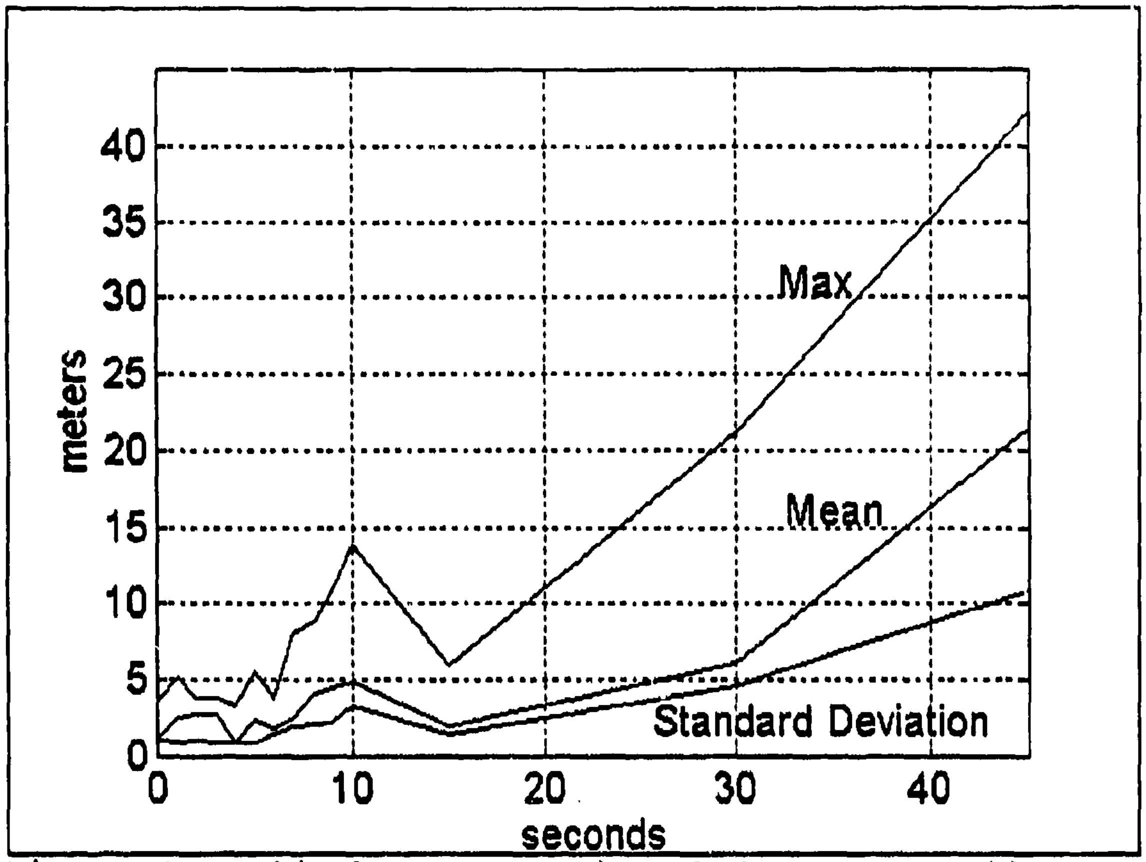


Figure 39 Graphical Representation Of The Data In Table VI Showing The Effects Of Time On The Accuracy Of DGPS Corrections.

VI. CONCLUSION

This thesis reviews the major contemporary electronic navigation systems available and demonstrates that GPS offers the best accuracies and world wide coverage on a realtime basis. The user segmont of GPS is explained, and its capabilities and deficiencies discussed. Differential GPS is presented and shown to be able to achieve enhanced accuracies sufficient to significantly contribute to this project.

An extensive review of available GPS receivers was conducted and Motorola's PVT-6 was found to combine all the benefits of GPS in a small, lightweight, low power and inexpensive package. Motorola's proprietary binary interface protocol allows the user great flexibility with little sacrifice of convenience. In addition, the standardized RS-232 serial communications port enables easy access to a variety of different systems. All things considered, it is the best choice over all other GPS receivers for this project.

The PVT-6's reacquisition times compare favorably with the manufacturer's claims. When two of these receivers were used in a DGPS mode with a zero baseline, the results easily confirmed nominal results of one to three meters desired accuracy. The most encouraging result is the relative immunity of DGPS accuracy to short delays (one to seven seconds) of the pseudorange correction inputs.

The success of this project's ultimate goal: to perform an autonomous landing, will depend on improving processing speeds and accuracies. A few areas where further investigation into GPS could contribute to this end are listed below.

- Conduct dynamic tests to confirm expected nominal accuracies using DGPS.
- Conduct additional data collection to determine if the accuracy of DGPS or reacquisition time is affected by the time of day or by the weather.
- Collect more data to determine if psuedorange correction latency steadily degrades or has anomalies (wherein the distance error actually improves with time) as Figure 39 suggests at the 15 second mark.
- Determine the effects of the aircraft's RF environment on the GPS performance.
- Investigate whether programming in assembly or some other language might improve processing speeds.
- Determine if the PVT-6's capability to compute two dimensional position by constraining altitude (relying solely on barometric altitude sensor data for altitude inputs) might improve robustness of GPS satellite tracking.
- Write a serial port driver that is independent of the PCL-744 multiple serial port I/O card in order to provide flexibility in using the PVT-6 and possibly reduce the total hardware weight through the elimination of the PCL-744 itself.
- Write a mission initialization interface routine to setup up the PVT-6 receivers.
- Write a post-processing routine to interpret and present the recorded results in a desired format.
- Write a real time graphical interface to plot the coordinates on a map. Also include either a separate window or color coded routine to be able to see current altitude.

GPS has revolutionized electronic navigation. The airline industry and other nations decry the U.S. DOD's introduction of Selective Availability. They are actively lobbying the U.S. government to force the DOD to turn it off and leave it off. It is an uphill battle but not a hopeless one. In time, as civilian applications become common place in our society, the DOD may acquiesce and turn it off. If that happens, GPS accuracies achieved in this project will undoubtedly improve and may warrant reconsideration in the way they are used in the Kalman Filter that integrates data from all the sources.

APPENDIX A: COORDINATE TRANSFORMATIONS

A. GEODETIC COORDINATE TRANSFORMATION TO ECEF COORDINATES

Given the geodetic coordinates longitude (λ), latitude (ϕ) and altitude (h), the Earth Centered Earth Fixed coordinates (X,Y,Z) are (Leick, 1990, p. 184-185):

$$X = (N + h) \cos\phi \cos\lambda$$
$$Y = (N + h) \cos\phi \sin\lambda$$
$$Z = [N (1 - e^2) + h] \sin\phi$$

where h is the height above the ellipsoid and

$$e^2 = 2f - f^2$$
$$f = \frac{a - b}{a}$$
$$a = \textit{semi-major axis}$$
$$b = \textit{semi-minor axis}$$
$$N = \frac{a}{\sqrt{1 - e^2 \sin\phi}}$$

B. ECEF COORDINATE TRANSFORMATION TO GEODETIC COORDINATES

Given the ECEF coordinates (X,Y,Z), the Geodetic coordinates (λ,ϕ,h) are (Leick, 1990, p. 184-185):

$$\phi = \arctan\left[\frac{1}{1 - e^2}\ \frac{Z}{\sqrt{X^2 + Y^2}}\right]$$

$$\lambda = \arctan\frac{Y}{X}$$

$$h = \frac{\sqrt{X^2 + Y^2}}{\cos\phi} - N$$

APPENDIX B: EPHEMERIS ALGORITHM

The following parameters are provided continuously for each satellite of the GPS constellation. Their exact placement and description within the ephemeris message is defined by the *Interface Control Document, ICD-GPS-200* as published by Rockwell International Space Systems Division. Any serious manipulation of the ephemeris algorithm using GPS data requires careful consideration of this document. There are several important, yet subtle, facts about the time tagging and the corresponding validity of the Issue of Data messages.

M_0	Mean Anomaly at Reference Time
Δn	Mean Motion Difference from Computed Value
e	Eccentricity
$(A)^{1/2}$	Square Root of the Semi-Major Axis
Ω_0	Longitude of Ascending Node of Orbit Plane at Weekly Epoch
i_0	Inclination Angle at Reference Time
ω	Argument of Perigee
OMEGADOT	Rate of Right Ascension
IDOT	Rate of Inclination Angle
C_{uc}	Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude
C_{us}	Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude

C_{rc}	Amplitude of the Cosine Harmonic Correction Term to the Orbit Radius
C_{rs}	Amplitude of the Sine Harmonic Correction Term to the Orbit Radius
C_{ic}	Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination
C_{is}	Amplitude of the Cosine Harmonic Correction Term to the Angle of Inclination
t_{oe}	Reference Time Ephemeris
IODE	Issue of Data (Ephemeris)

The following tables list the number of bits, the location within words three through ten (omitting the parity bits), the scale factor of the LSB, the range and the units for each of subframes two (Table VII) and three (Table VIII). Parameters indicated with an asterisk are to be two's complimented with the sign bit in the MSB. Unless otherwise listed in the value range column, effective range is the maximum range attainable with the indicated bit allocation and scalr factor.

The following elements of the ephemeris algorithm are listed in order of expected solution. All the required information is given or may be found using the above from the GPS ephemeris message.

$\mu = 3.986005 \times 10^{14} \ \frac{\text{meters}^3}{\text{sec}^2}$	WGS 84 value of the earth's universal gravitational parameter

Table VII BIT ALLOCATION AND SCALING FACTORS FOR WORDS THREE THROUGH TEN OF SUBFRAME TWO. (ROCKWELL, 1987, PP. 60-81)

Parameter	No. of bits	Location of bits	LSB scale factor	Value range	Units
IODE	8	1-8			see ICD-GPS-200
C_{rs}	16*	9-24	2^{-5}		meters
Δn	16*	25-40	2^{-43}		semi-circles/sec
M_0	32*	41-72	2^{-31}		semi-circles
C_{uc}	16*	73-88	2^{-29}		radians
e	32	89-120	2^{-33}	0.03	dimensionless
C_{us}	16*	121-136	2^{-29}		radians
$(A)^{1/2}$	32	137-168	2^{-19}		$\text{meters}^{1/2}$
t_{oe}	16	169-184	2^{4}	604,784	seconds
other	8	185-192			see ICD-GPS-200

**Parameters indicated are signed integers. All others are unsigned.*

$$\Omega = 7.2921151467 \times 10^{-5} \ \frac{\text{rad}}{\text{sec}}$$ WGS 84 value of the earth's rotation rate

$$A = (\sqrt{A})^2$$ Semi-major axis

$$n_0 = \sqrt{\frac{\mu}{A^3}} \ \frac{\text{rad}}{\text{sec}}$$ Computed mean motion

$$t_k = t - t_{oe}$$ Time from ephemeris reference epoch

In the above equation, t is GPS system time at time of transmission (i.e., GPS time corrected for transit time

Table VIII BIT ALLOCATION AND SCALING FACTORS FOR WORDS THREE THROUGH TEN OF SUBFRAME THREE. (ROCKWELL, 1987, PP. 60-81)

Parameter	No. of bits	Bit location	LSB scale factor	Value range	Units
C_{ic}	16*	1-16	2^{-29}		radians
Ω_0	32*	17-48	2^{-31}		semi-circles
C_{is}	16*	49-64	2^{-29}		radians
i_0	32*	65-96	2^{-31}		semi-circles
C_{rc}	16*	97-112	2^{-5}		meters
ω	32*	113-144	2^{-31}		semi-circles
OMEGADOT	24*	145-168	2^{-43}		semi-circles/sec
IODE	8	169-176			see ICD-GPS-200
IDOT	14*	177-190	2^{-43}		semi-circles/sec
other	2	191-192			see ICD-GPS-200

**Parameters indicated are signed integers. All others are unsigned.*

range/speed of light. Furthermore, t_k shall be the actual total time difference between the time t and the epoch time t_{oe}, and must account for beginning or end of week crossovers. That is, if t_k is greater than 302,400 seconds, subtract 604,800 seconds from t_k. If t_k is less than 302,400 seconds, add 604,800 seconds to t_k.

$n = n_0 + \Delta n$	Corrected mean motion
$M_k = M_0 + nt_k$	Mean anomaly

The next equation solves for the eccentric anomaly (E_k) by iteration. M_k and e are known from above.

Let $E_1 = \frac{M_k}{1 - e}$

Then $M_1 = E_1 - e \sin E$

$\Delta_1 = \frac{M_k - M_1}{1 - e \cos E}$

$E_2 = E_1 + \Delta_1$

If $M_2 \approx M_k$, then $E_2 = E_k$ otherwise repeat and compare again. (Tattersfield, 1984, p. 36)

Having found E_k, the rest of the solution is simply solving equations.

$v_k = \arctan\left[\frac{\sqrt{1 - e^2}\sin E_k}{\cos E_k - e}\right]$ True anomaly

$\Phi = v_k + \omega$ Argument of latitude

Second harmonic perturbations:

$\delta u_k = C_{us} \sin 2\Phi_k + C_{uc} \cos 2\Phi_k$	Argument of lat correction
$\delta r_k = C_{rc} \cos 2\Phi_k + C_{rs} \sin 2\Phi_k$	Radius correction
$\delta i_k = C_{ic} \cos 2\Phi_k + C_{is} \sin 2\Phi_k$	Correction to inclination

$u_k = \Phi_k + \delta r_k$ Corrected argument of latitude

$r_k = A(1 - e \cos E_k) + \delta r_k$ Corrected radius

$i_k = i_0 + \delta i_k + (\mathrm{IDOT})\, t_k$ Corrected inclination

$$\left\{\begin{array}{l} x'_k = r_k \cos u_k \\ y'_k = r_k \sin u_k \end{array}\right\}$$ Positions in orbital plane

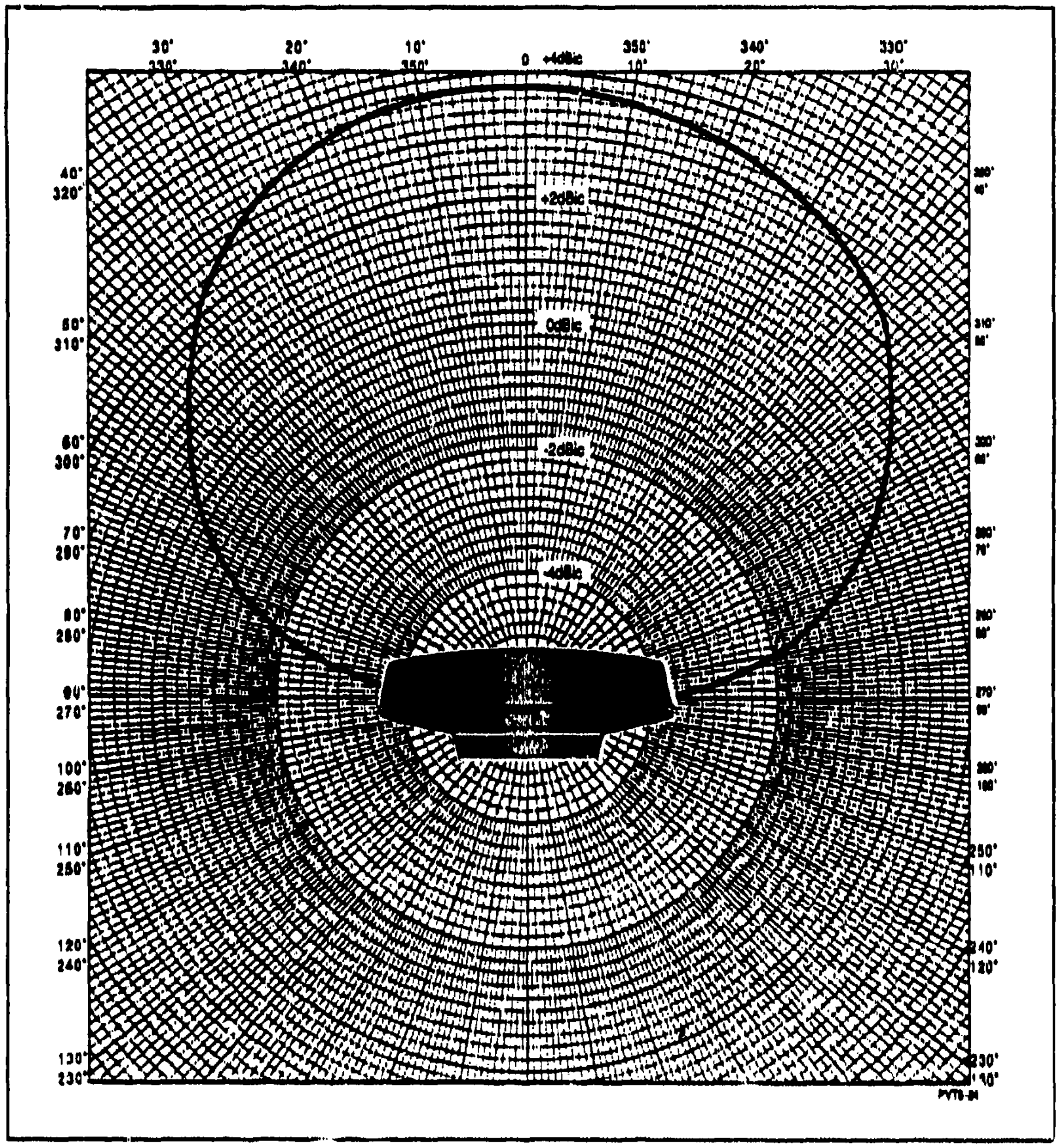
$$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e)\, t_k - \dot{\Omega}_e t_{oe}$$ Corrected longitude of ascending node

$$\left\{\begin{array}{l} x_k = x'_k \cos\Omega_k - y'_k \cos i_k \sin\Omega_k \\ x_k = x'_k \sin\Omega_k - y'_k \cos i_k \cos\Omega_k \\ z_k = y'_k \sin i_k \end{array}\right\}$$ Earth-Centered, Earth-Fixed (ECEF) coordinates

APPENDIX C: PVT-6 SPECIFICATIONS

Parameter	Description
Dynamics	• Velocity 1000 Knots (515 m/sec) • Acceleration 4 g • Jerk $5 m/s^3$
Antenna	• Active microstrip patch Antenna Module • Powered by Receiver Module (5 Vdc @ 25 mA)
Antenna to Receiver Interconnection	• Single coaxial cable (6dB max loss at L1; 1575.42 MHz) • Typical length with RG58 coaxial cable; 20 feet (6 meters)
Serial Output	• RS232C Interface
Accuracy	• Less than 25 meters, SEP (without SA) * DoD may invoke Selective Availability (SA), potentially degrading accuracy to 100 meters (2d RMS)
Altitude	• 60,000 feet (18 Km) (maximum)
Operating Temperature	• Active Antenna -40°C to +100°C (-40°F to +212°F) • Receiver Module -30°C to +80°C (-22°F to 176°F)
Humidity	• 95% non-condensing +30°C to +60°C (+86°F to +140°F)
Physical Dimensions	• Receiver PCB 3.94 X 2.76 X 0.65 in (100 X 70 X 16.5 mm) • Antenna Module 4.01 (dia) X 0.89 in (102 (dia) X 22.6 mm)
Weight	• Receiver and Housing 4.5 oz (128 g) • Antenna Module 4.8 oz (136.2 g)
Switched Power	• 9 to 16 Vdc or • 5 ±0.25 Vdc
"Keep-Alive" BATT Power	• 4.75 - 16 Vdc; 0.3 mA
Power Consumption (typical)	• 1.3 W @ 5 Vdc input • 1.8 W @ 12 Vdc input
Receiver Interconnection	• AMP 10 pin, #104369-4 on Receiver Module - RS232C - Power Input • OSX connector on Receiver and Antenna
MTBF (Mean Time Before Failure)	• 55,000 hours (estimated)

APPENDIX D: PVT-6 ANTENNA GAIN



APPENDIX E: SOFTWARE DRIVER SOURCE CODE

A. GPSAN.C

```
/*  GPSAN.C

        Reads ascii position data "@@Ba" from a file, converts it
        it to binary and places the scaled and formatted messages
        in an array of structures.

        Input  is the name of the ascii file without the extension.
        Output is a file with the analyzed data from GPSCRNCH.H.
*/

#include  <string.h>
#include  <stdlib.h>
#include  <stdio.h>
#include  "c:\borlandc\twite\gpstruct.h"
#include  "c:\borlandc\twite\gpsconv.h"
#include  "c:\borlandc\twite\gpscrnch.h"
#include  "c:\borlandc\twite\gpsfun.h"

void main(void)
{
    #define  COL      130
    #define  NEWCOL   66

    char name[40], path[30] = "c:\\borlandc\\etwork\\";
    int      i,j,k,m,n,s,master,slave, ROW;
    ONEBYTE  first, second, third, inbuff[COL];
    ONEBYTE  **mbuff, **sbuff, **mbuffbin, **sbuffbin;
    struct   T_POS_CHAN_STATUS  *marst, *sarst, pos_chan;
    FILE *fpname;

    printf("\nEnter the input filename (without the extension(??.ext):  ");
    scanf("%s",name);
    strcat(path, name);
    strcat(path, ".txt");
    printf("%s",path);
    fpname = fopen(path,"r");

    printf("\nEnter the number of rows (individual messages):  ");
    scanf("%d",&ROW);
    printf("\n");
```

```
k = 0;
m = 0;
s = 0;

/* Create mbuff and sbuff.
*/
mbuff = calloc( ROW, sizeof( ONEBYTE * ));
sbuff = calloc( ROW, sizeof( ONEBYTE * ));
for ( i=0; i<ROW; i++)
{
    mbuff[i] = calloc( COL, sizeof( ONEBYTE ));
    sbuff[i] = calloc( COL, sizeof( ONEBYTE ));
};

while (1)
{
    /* Read in file to ascii arrays mbuff and sbuff.
    */
    if ( k >= 70 )
        break;
    first  = fgetc(fpname);
    if ( first == '@' )
    {
        second = fgetc(fpname);
        if (second == '@' )
        {
            third  = fgetc(fpname);
            /* Store inbuff array in either mbuff or sbuff.
            */
            if (third == '@')
            {
                fread( sbuff[s], sizeof(ONEBYTE), COL, fpname );
                k = 0;
                s++;
            }
            else
            {
                mbuff[m][0] = third;
                for  (i=1; i<COL; i++)
                    mbuff[m][i] = fgetc(fpname);
                k = 0;
                m++;
            };
        };
    };
k++;
}; /* end while */
fclose(fpname);

master = m;
slave  = s;
```

```
/* Create mbuffbin and sbuffbin.
*/
mbuffbin = calloc( master, sizeof( ONEBYTE * ));
for ( i=0; i<master; i++)
    mbuffbin[i] = calloc( NEWCOL, sizeof( ONEBYTE ));
sbuffbin = calloc( slave, sizeof( ONEBYTE  * ));
for ( i=0; i<slave; i++)
    sbuffbin[i] = calloc( NEWCOL, sizeof( ONEBYTE ));

/* Convert hex buffer to binary buffer.
*/
for (i = 0; i < master; i++ )
{
    mbuffbin[i][0] = mbuff[i][0];
    mbuffbin[i][1] = mbuff[i][1];
    k = 2;
    for ( j = 2; j < COL; j+=2)
    {
        mbuffbin[i][k] = (Hex(mbuff[i][j]) << 4) | Hex(mbuff[i][j+1]);
        k += 1;
    };
};

for (i = 0; i < slave; i++ )
{
    sbuffbin[i][0] = sbuff[i][0];
    sbuffbin[i][1] = sbuff[i][1];
    k = 2;
    for ( j = 2; j < COL; j+=2)
    {
    sbuffbin[i][k] = (Hex(sbuff[i][j]) << 4) | Hex(sbuff[i][j+1]);
    k += 1;
    };
};

/* Free mbuff and sbuff memory.
*/
for (i=0; i<ROW; i++)
{
    free(mbuff[i]);
    free(sbuff[i]);
};
free(mbuff);
free(sbuff);

/* Temporarily store mbuffbin in a file "mtemp.bin" and free memory.
*/
fpname = fopen( "mtemp.bin", "wb" );
for (i=0; i<master; i++)
    fwrite(mbuffbin[i], sizeof(ONEBYTE), NEWCOL, fpname);
for (i=0; i<master; i++)
```

```
    free(mbuffbin[i]);
free(mbuffbin);
fclose(fpname);

/* Temporarily store sbuffbin in a file "stemp.bin" and free memory.
*/
fpname = fopen( "stemp.bin", "wb" );
for (i=0; i<slave; i++)
    fwrite(sbuffbin[i], sizeof(ONEBYTE), NEWCOL, fpname);
for (i=0; i<slave; i++)
    free(sbuffbin[i]);
free(sbuffbin);
fclose(fpname);

/* Create an array of structures.
*/
marst = calloc( master, sizeof( struct T_POS_CHAN_STATUS ));
sarst = calloc( slave , sizeof( struct T_POS_CHAN_STATUS ));

fpname = fopen( "mtemp.bin", "rb");
for ( i = 0; i < master; i++)
{
    fread(inbuff, sizeof(ONEBYTE), NEWCOL, fpname);
    Convert_Pos_Chan_Status( inbuff );
    marst[i] = pos_chan;
};
fclose( fpname );
remove("mtemp.bin");
fpname = fopen( "stemp.bin", "rb");
for ( i = 0; i < slave; i++)
{
    fread(inbuff, sizeof(ONEBYTE), NEWCOL, fpname);
    Convert_Pos_Chan_Status( inbuff );
    sarst[i] = pos_chan;
};
fclose( fpname );
remove("stemp.bin");

Ave_Max_Stdev(marst, sarst, master);

}; /* end main */
```

B. GPSCONV.H

```
/* GPSCONV.H

        Convert_Ephemeris()
        Convert_Pos_Chan_Status()
        Convert_xDOP()
        Convert_Sat_Range_Data()
        Convert_Pos_Chan_Status_Ext()
*/

#ifndef  _GPSCONV_H
#define  _GPSCONV_H

#include  <math.h>

#ifndef  _GPSTRUCT_H
#include  "c:\borlandc\twite\gpstruct.h"
#endif

/*****************************************************************
---------------------------------------------------------------
    Convert_Ephemeris()

      This procedure converts the raw binary ephemeris data
      into a representation usable for post-processing
      applications.  The equations coded in this procedure
      are taken from the ICD-GPS-200 (the GPS NAVSTAR ICD).

      Input  is an array with the binary ephemeris message.
             Although it is unscaled, it is in the array
             format of the ICD-GPS-200 format.
      Output is a structure with the scaled and formatted
             ephemeris message.
*/

void Convert_Ephemeris( ONEBYTE hex_ephemeris[][3] )
  {
  /* Sign extends a neg hex byte into its TWOBYTE (integer) equivalent
  */
  #define MACRO_SET_UPPER(word_to_set) \
                  if ( word_to_set & 0x0080 )  word_to_set |= 0xFF00 ;

  /* This MACRO takes two hex bytes and merges them into a single
     TWOBYTE (integer) value.
  */
  #define MACRO_MAKE_EPH_WORD(e_wd, w1, b1, w2, b2) \
        e_wd = ( ( ( UNSIGNED_TWOBYTE ) hex_ephemeris[w1][b1] << 8 ) | \
                  ( UNSIGNED_TWOBYTE ) hex_ephemeris[w2][b2] );
```

```
UNSIGNED_TWOBYTE word1, word2; /* Temporary storage for words built. */
struct T_FLOAT_EPHEMERIS *eph ;

eph = malloc(sizeof( struct T_FLOAT_EPHEMERIS) );

/* Time of ephemeris
*/
MACRO_MAKE_EPH_WORD(word1,  15, 0, 15, 1 );
eph->toe = (double) ( ((UNSIGNED_FOURBYTE) word1) * 16 ) ;

/* Mean anomaly at reference time ( m0 ).
*/
MACRO_MAKE_EPH_WORD(word1,  9, 2, 10, 0 );
MACRO_MAKE_EPH_WORD(word2, 10, 1, 10, 2 );
eph->m0 = ( ( ( FOURBYTE ) word1 << 16 ) | word2 ) * PITWOM31;

/* Mean motion difference from computed value ( delta_n ).
*/
MACRO_MAKE_EPH_WORD(word1,  9, 0, 9, 1 );
eph->delta_n = ( TWOBYTE ) word1 * PITWOM43;

/* Eccentricity ( e ).
*/
MACRO_MAKE_EPH_WORD(word1,  11, 2, 12, 0 );
MACRO_MAKE_EPH_WORD(word2,  12, 1, 12, 2 );
eph->e = ( ( ( UNSIGNED_FOURBYTE ) word1 << 16 ) | word2 ) * TWOM33;

/* Square root of the semi-major axis ( sqrt_a ).
*/
MACRO_MAKE_EPH_WORD(word1,  13, 2, 14, 0 );
MACRO_MAKE_EPH_WORD(word2,  14, 1, 14, 2 );
eph->sqrt_a = ( ( ( UNSIGNED_FOURBYTE ) word1 << 16 ) | word2 ) * TWOM19;

/* Longitude of ascending mode of orbit plane at
   weekly epoch ( omega_0 ).
*/
MACRO_MAKE_EPH_WORD(word1,  16, 2, 17, 0 );
MACRO_MAKE_EPH_WORD(word2,  17, 1, 17, 2 );
eph->omega_0 = ( ( ( FOURBYTE ) word1 << 16 ) | word2 ) * PITWOM31;

/* Inclination angle at reference time ( i0 ).
*/
MACRO_MAKE_EPH_WORD(word1,  18, 2, 19, 0 );
MACRO_MAKE_EPH_WORD(word2,  19, 1, 19, 2 );
eph->i0 = ( ( ( FOURBYTE ) word1 << 16 ) | word2 ) * PITWOM31;

/* Argument of perigee ( w ).
*/
MACRO_MAKE_EPH_WORD(word1,  20, 2, 21, 0 );
MACRO_MAKE_EPH_WORD(word2,  21, 1, 21, 2 );
eph->w = ( ( ( FOURBYTE ) word1 << 16 ) | word2 ) * PITWOM31;
```

```
/* Rate of right ascension ( omega_dot ).
*/
word1 = ( UNSIGNED_TWOBYTE ) hex_ephemeris[22][0];
MACRO_SET_UPPER(word1) ;  /* Set upper bits if negative. */
MACRO_MAKE_EPH_WORD(word2, 22, 1, 22, 2 );
eph->omega_dot = ( ( ( FOURBYTE ) word1 << 16 ) | word2 ) * PITWOM43;

/* Rate of inclination angle ( i_dot ).
*/
MACRO_MAKE_EPH_WORD(word1, 23, 1, 23, 2 ) ;
word1 = word1 >> 2 ;
if ( word1 & 0x2000 )    /* Set upper bits if negative. */
   word1 |= 0xC000;
eph->i_dot = ( TWOBYTE ) word1 * PITWOM43;

/* Amplitude of the cosine harmonic correction term to the
   argument of latitude (cuc).
*/
MACRO_MAKE_EPH_WORD(word1, 11, 0, 11, 1 );
eph->cuc = ( TWOBYTE ) word1 * TWOM29;

/* Amplitude of the sine harmonic correction term to the
   argument of latitude (cus).
*/
MACRO_MAKE_EPH_WORD(word1, 13, 0, 13, 1 );
eph->cus = ( TWOBYTE ) word1 * TWOM29;

/* Amplitude of the cosine harmonic correction term to the
   orbit radius (crc).
*/
MACRO_MAKE_EPH_WORD(word1, 20, 0, 20, 1 );
eph->crc = ( TWOBYTE ) word1 * TWOM5;

/* Amplitude of the sine harmonic correction term to the
   orbit radius (crs).
*/
MACRO_MAKE_EPH_WORD(word1, 8, 1, 8, 2 );
eph->crs = ( TWOBYTE ) word1 * TWOM5;

/* Amplitude of the cosine harmonic correction term to the
   angle of inclination (cic).
*/
MACRO_MAKE_EPH_WORD(word1, 16, 0, 16, 1 );
eph->cic = ( TWOBYTE ) word1 * TWOM29;

/* Amplitude of the sine harmonic correction term to the
   angle of inclination (cis).
*/
MACRO_MAKE_EPH_WORD(word1, 18, 0, 18, 1 );
eph->cis = ( TWOBYTE ) word1 * TWOM29;
```

```
/* Estimated group delay differential ( tgd ).
*/
word1 = ( UNSIGNED_TWOBYTE ) hex_ephemeris[4][2];
MACRO_SET_UPPER(word1) ;   /* Set upper bits if negative. */
eph->tgd = ( TWOBYTE ) word1 * TWOM31;

/* Clock data reference time ( toc ).
*/
MACRO_MAKE_EPH_WORD(word1,  5, 1, 5, 2 );
eph->toc = word1 * 16.0;

/* Polynomial coefficient ( af2 ).
*/
word1 = ( UNSIGNED_TWOBYTE ) hex_ephemeris[6][0];
MACRO_SET_UPPER(word1) ;   /* Set upper bits if negative. */
eph->af2 = ( TWOBYTE ) word1 * TWOM55;

/* Polynomial coefficient ( af1 ).
*/
MACRO_MAKE_EPH_WORD(word1,  6, 1, 6, 2 );
eph->af1 = ( TWOBYTE ) word1 * TWOM43;

/* Polynomial coefficient ( af0 ).
*/
word1 = ( UNSIGNED_TWOBYTE ) hex_ephemeris[7][0];
MACRO_SET_UPPER(word1) ;   /* Set upper bits if negative. */
MACRO_MAKE_EPH_WORD(word2,  7, 1, 7, 2 ) ;
word2 = word2 & 0xFFFC ; /* clear 2 bits */
eph->af0 = ( ( ( FOURBYTE ) word1 << 16 ) | word2 ) * TWOM3125;

/* Semi-major axis ( a ).
*/
eph->a = eph->sqrt_a * eph->sqrt_a;

/* Corrected mean motion ( n ).
*/
eph->n = SQRMU / ( eph->a * eph->sqrt_a ) + eph->delta_n;

/* Square root of eccentricity^2 subtracted from 1 ( esqrt ).
*/
eph->e_sqrt = sqrt ( 1.0 - eph->e * eph->e );

/* Omega_dot - WE ( wk1 ).
*/
eph->wk1 = eph->omega_dot - WE;

/* Omega0 - WE * toe ( wk0 ).
*/
eph->wk0 = eph->omega_0 - WE * eph->toe;

/* Sine w ( sin_w ).
```

```
  */
  eph->sin_w = sin ( eph->w );

  /* Cosine w ( cos_w ).
  */
  eph->cos_w = cos ( eph->w );

  /* Square root of semi-major axis * eccentricity * FCONST.
  */
  eph->sqrta_e_fconst = eph->sqrt_a * eph->e * FCONST;

  /* Issue of data, ephemeris.
  */
  eph->iode = hex_ephemeris[23][0];

return *eph;
};

/*****************************************************************
*******************************************************************
    Convert_Pos_Chan_Status()

        This procedure parses and decodes a position/status/data
    record received from the Motorola PVT-6. The flow of this
    procedure is as follows :
        - pass in the BINARY buffer containing 66 bytes
        - read the BINARY data from the buffer and convert directly
          into the final data structure.

        Input is a buffer of type ONEBYTE with a single binary
            position/status/data message.
        Output is the structure with the scaled and formatted
            position/status/data message.
*/

struct T_POS_CHAN_STATUS Convert_Pos_Chan_Status(ONEBYTE pos[])
     {
     struct  T_POS_CHAN_STATUS  pos_chan;
     UNSIGNED_ONEBYTE  i, j ;
     UNSIGNED_ONEBYTE  tempchar ;
     UNSIGNED_FOURBYTE tempu4byte ;
     FOURBYTE temps4byte ;
     UNSIGNED_FOURBYTE nsecs ;
     double degrees, minutes, seconds ;

     /* Read and scale the of the data.
     */
     pos_chan.month = pos[2] ;
     pos_chan.day   = pos[3] ;

     tempchar = pos[4] ;
```

```
pos_chan.year = ( tempchar << 8 ) + pos[5] ;

pos_chan.hours    = pos[6] ;
pos_chan.minutes  = pos[7] ;

tempchar   = pos[8] ; /* integer seconds */
tempu4byte = pos[9] ;
tempu4byte = ( tempu4byte << 8 ) + pos[10] ;
tempu4byte = ( tempu4byte << 8 ) + pos[11] ;
tempu4byte = ( tempu4byte << 8 ) + pos[12] ;
pos_chan.seconds = (double) tempchar + \
         ( ( (double) tempu4byte ) / 1.0E+9 );

temps4byte = pos[13] ;
temps4byte = ( temps4byte << 8 ) + pos[14] ;
temps4byte = ( temps4byte << 8 ) + pos[15] ;
temps4byte = ( temps4byte << 8 ) + pos[16] ;
degrees = (double) temps4byte * MSECS_TO_DEGREES ;

pos_chan.latitude.degrees = (TWOBYTE) degrees ;
if ( degrees < 0 )
degrees = fabs ( degrees ) ;
minutes =  ( degrees - (TWOBYTE) degrees ) * 60.0 ;
pos_chan.latitude.minutes = (TWOBYTE) ( minutes ) ;
pos_chan.latitude.seconds = ( minutes - (TWOBYTE) minutes ) * 60.0 ;

temps4byte = pos[17] ;
temps4byte = ( temps4byte << 8 ) + pos[18] ;
temps4byte = ( temps4byte << 8 ) + pos[19] ;
temps4byte = ( temps4byte << 8 ) + pos[20] ;
degrees = (double) temps4byte * MSECS_TO_DEGREES ;

pos_chan.longitude.degrees = (TWOBYTE) degrees ;
if ( degrees < 0 )
degrees = fabs ( degrees ) ;
minutes = ( degrees - (TWOBYTE) degrees ) * 60.0 ;
pos_chan.longitude.minutes = (TWOBYTE) ( minutes ) ;
pos_chan.longitude.seconds = ( minutes - (TWOBYTE) minutes ) * 60.0 ;

temps4byte = pos[21] ;
temps4byte = ( temps4byte << 8 ) + pos[22] ;
temps4byte = ( temps4byte << 8 ) + pos[23] ;
temps4byte = ( temps4byte << 8 ) + pos[24] ;
pos_chan.datum_height = (double) temps4byte / 100.0 ;

temps4byte = pos[25] ;
temps4byte = ( temps4byte << 8 ) + pos[26] ;
temps4byte = ( temps4byte << 8 ) + pos[27] ;
temps4byte = ( temps4byte << 8 ) + pos[28] ;
pos_chan.msl_height = (double) temps4byte / 100.0 ;
```

```
    tempchar = pos[29] ;
    pos_cnan.velocity = (double) ( ( tempchar << 8 ) + pos[30] ) / 100.0 ;

    tempchar = pos[31] ;
    pos_chan.heading = (double) ( ( tempchar << 8 ) + pos[32] ) / 10.0 ;

    tempchar = pos[33] ;
    pos_chan.current_dop = (double) ( ( tempchar << 8 ) + pos[34] ) / 10.0;

    pos_chan.dop_type     = pos[35] ;
    pos_chan.visible_sats = pos[36] ;
    pos_chan.sats_tracked = pos[37] ;

    j=0;
    for (i = 0; i < NUM_CHANNELS; i++)
      {
      pos_chan.channel[i].svid     = pos[38 + j];
      pos_chan.channel[i].mode     = pos[39 + j];
      pos_chan.channel[i].strength = pos[40 + j];
      pos_chan.channel[i].flags    = pos[41 + j];
      j+=4;
      }
    pos_chan.rcvr_status = pos[62];

    return  pos_chan;
    };

#endif   /* _GPSCONV_H */
```

C.   GPSCRNCH.H

```
/*  GPSCRNCH.H

          Used to crunch the position data to determine the mean,
      maximum, and standard deviation of the position error
      obtained by DGPS. Geodetic coordinates are converted to
      ECEF coordinates and the distance formula is used to compute
      the distance error.
          Input is an array of structures that contain the scaled
              and formatted position/chan/status message data.
              The user is asked to supply the output file name
              without the extension.
          Output is a file with the distance error for each data
              point, and the mean, max and standard deviation of
              all the data points.
*/

#ifndef  _GPSCRNCH_H
#define  _GPSCRNCH_H

#include  <math.h>
#include  <stdio.h>
#include  <string.h>

#ifndef  _GPSTRUCT_H
#include  "c:\borlandc\twite\gpstruct.h"
#endif

FILE *fp;

void Ave_Max_Stdev(struct T_POS_CHAN_STATUS *mast, \
                   struct T_POS_CHAN_STATUS *slav, int master)
  {
  int     i, count=0, start;
  char name[40], outpath[30]="c:\\borlandc\\twite\\";
  double   mlat, mlon, malt, slat, slon, salt, mN, sN, mean, distsum=0;
  double  mx, my, mz, sx, sy, sz, dist[200], max=-1.0, stdev, distsum2=0;

  printf("Enter the output filename without the extension (???.txt):  ");
  scanf("%s",name);
  strcat(outpath, name);
  strcat(outpath, ".out");
  fp = fopen(outpath,"w");
  fprintf(fp,"\n%s\n",outpath);

  printf("Enter the starting data point number:  ");
  scanf("%d",&start);
```

```
for (i=start; i<master; i++)
{
    /* Test to ensure the seconds match */
    if ( (ONEBYTE) mast[i].seconds == (ONEBYTE) slav[i].seconds  )
    {
    /* Convert to mx, my, mz, sx, sy, sz.
    */
    mlat = (mast[i].latitude.seconds * (1/60) + \
           mast[i].latitude.minutes) * (1/60) + mast[i].latitude.degrees;
    mlon = (mast[i].longitude.seconds * (1/60) + \
           mast[i].longitude.minutes) * (1/60) + mast[i].longitude.degrees;
    malt = mast[i].datum_height;

    slat = (slav[i].latitude.seconds * (1/60) + \
           slav[i].latitude.minutes) * (1/60) + slav[i].latitude.degrees;
    slon = (slav[i].longitude.seconds * (1/60) + \
           slav[i].longitude.minutes) * (1/60) + slav[i].longitude.degrees;
    salt = slav[i].datum_height;

    mN   = SEMIMAJOR_AXIS / sqrt( 1 - E_SQUARED * sin(mlat) );
    sN   = SEMIMAJOR_AXIS / sqrt( 1 - E_SQUARED * sin(slat) );

    mx   = ( mN + malt ) * cos(mlat) * cos(mlon);
    my   = ( mN + malt ) * cos(mlat) * sin(mlon);
    mz   = ( mN * ( 1 - E_SQUARED ) + malt) * sin(mlat);

    sx   = ( sN + salt ) * cos(slat) * cos(slon);
    sy   = ( sN + salt ) * cos(slat) * sin(slon);
    sz   = ( sN * ( 1 - E_SQUARED ) + salt) * sin(slat);

    dist[i] = sqrt( (mx-sx)*(mx-sx) + (my-sy)*(my-sy) + (mz-sz)*(mz-sz) );

    if ( dist[i] > 1000 )
    {
        fprintf(fp,"\nMtime = %2d:%10.7f <-> %2d:%10.7f = Stime   \
                Dist #%3d = XXXXX",mast[i].minutes, mast[i].seconds, \
                slav[i].minutes, slav[i].seconds, i);
        continue;
    };

    /* Compute average, max and standard deviation.
    */
    count++;
    if ( dist[i] > max ) max = dist[i];
    distsum  += dist[i];
    distsum2 += dist[i] * dist[i];

    /* Print out the results to a file.
    */
    fprintf(fp,"\nMtime = %2d:%10.7f <-> %2d:%10.7f = Stime   \
            Dist #%3d = %5.2f",mast[i].minutes, mast[i].seconds, \
            slav[i].minutes, slav[i].seconds, i, dist[i]);
    }
    else
    {
    fprintf(fp,"\nMtime = %2d:%10.7f <-> %2d:%10.7f = Stime   \
            Dist #%3d = XXXXX",mast[i].minutes, mast[i].seconds, \
            slav[i].minutes, slav[i].seconds, i);
    };
};

/* Compute the mean and standard deviation.
*/
mean = distsum / count;
stdev = sqrt( (distsum2/count) - (mean * mean));

fprintf(fp, "\n\nMean  = %5.2f\nMax   = %5.2f\nStdev = %5.2f\n",\
        mean, max, stdev);

fclose(fp);

}; /* end Ave_Max_Stdev */

#endif   /* _GPSCRNCH_H */
```

D. GPSDEFIN.H

```
/*  GPSDEFIN.H

        Store most of the definitions here.
*/

#ifndef _GPSDEFIN_H
#define  _GPSDEFIN_H

/********* type definitions *********/

typedef char  ONEBYTE;  /* data type allowing for the use of  8 bits. */
typedef short TWOBYTE;  /* data type allowing for the use of 16 bits. */
typedef long  FOURBYTE; /* data type allowing for the use of 32 bits. */

typedef unsigned char  UNSIGNED_ONEBYTE;         /* unsigned  8 bits */
typedef unsigned short UNSIGNED_TWOBYTE;         /* unsigned 16 bits */
typedef unsigned long  UNSIGNED_FOURBYTE;        /* unsigned 32 bits */

/********* constant definitions *****/

#define PI       3.1415926535898      /* GPS value of PI          */
#define WE       7.2921151467E-5      /* WGS84 earth's rot rate (rads/sec) */

#define SQRMU   1.99649818432174E7  /* Square root of MU.          */
#define FCONST -4.442807633E-10     /* FCONST is used in relativistic
                     clock correction.  Defined on
                     page 73 of ICD-GPS-200.          */

#define      INVERSE_FLATTENING  298.257223563       /* WGS-84 unitless */
#define      FLATTENING          1/INVERSE_FLATTENING
#define      SEMIMAJOR_AXIS 6378137.0            /* WGS-84 meters   */
#define      E_SQUARED      2*FLATTENING-FLATTENING*FLATTENING

#define TWOM5      0.03125000000000    /* 2^-5  */
#define TWOM11      (TWOM5  / 64.0)
#define TWOM19      (TWOM11 / 256.0)
#define TWOM20      (TWOM19 /   2.0)
#define TWOM21      (TWOM20 /   2.0)
#define TWOM23      (TWOM21 /   4.0)
#define TWOM24      (TWOM23 /   2.0)
#define TWOM27      (TWOM24 /   8.0)
#define TWOM29      (TWOM23 /  64.0)
#define TWOM30      (TWOM29 /   2.0)
#define TWOM31      (TWOM30 /   2.0)
#define TWOM33      (TWOM31 /   4.0)
#define TWOM38      (TWOM33 /  32.0)
#define TWOM43      (TWOM38 /  32.0)
```

```
#define TWOM50      (TWOM43 / 128.0)
#define TWOM55      (TWOM50 /  32.0)
#define TWOM3125     (0.25 * TWOM31)

#define TWOP11      2048.0           /* 2^11  */
#define TWOP12      (TWOP11 * 2.0)
#define TWOP14      (TWOP12 * 4.0)
#define TWOP16      (TWOP14 * 4.0)

#define TWOP12_INTEGER  4096          /* 2^12 */
#define TWOP16_INTEGER  65536          /* 2^16  */

#define PITWOM31    (PI * TWOM31)
#define PITWOM43    (PI * TWOM43)

/********* Twite definitions ********/

#define     FALSE       0
#define     TRUE     1
#define     RX      0       /* receive */
#define     TX      1       /* send    */
#define     MGPS_PORT     3              /* master gps receiver port */
#define     SGPS_PORT     4        /* slave  gps receiver port */
#define     NUM_CHANNELS  6
#define     MSECS_TO_DEGREES    (( 1.0 / 1000.0 ) / 3600.0 )

#define  EPH_NUM_WORDS  24   /* at 3 bytes/word */

#endif /* _GPSDEFIN_H */
```

E. GPSDELAY.H

```
/* GPSDELAY.H

        Used to introduce an artificial delay in transmitting
        psuedorange corrections from the master to the slave.

        PR_delay() accepts a pointer of type PANDL to the buffer
                containing the psuedorange correction message
                (@@Ce...) and an integer number of seconds it
                is to be delayed. The output is to the PCL-744
                serial I/O card.
*/
#ifndef _GPSDELAY_H
#define _GPSDELAY_H

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <dos.h>
#include "c:\borlandc\twite\head-c.h"

#ifndef _GPSTRUCT_H
#include "c:\borlandc\twite\gpstruct.h"
#endif

#define MAXDELAY 300

static  PANDL   stor[MAXDELAY];     /* Ring buffer                 */
static  int  num_waiting = 0;   /* Number of messages in queue    */
static  int  next_stor   = 0;   /* Next storage location to be used */
static int   next_xmit   = 0;   /* Next message to be transmitted   */

void PR_delay( PANDL *Ce_msg, int delay )
   {
   PANDL          *ptr;

   /* Copy the contents of the buffer with the Ce_msg to a
      new buffer for holding during the delay.
   */
   ptr = malloc( sizeof(PANDL) );
   ptr->len = Ce_msg->len;
   ptr->ptr = calloc( ptr->len, sizeof(ONEBYTE) );
   memcpy( ptr->ptr, Ce_msg->ptr, Ce_msg->len );

   /* Store the structure of type PANDL in the ring buffer "stor[]".
   */
   if (num_waiting < MAXDELAY )
```

```
        {
        stor[next_stor] = *ptr;
        num_waiting++ ;
        next_stor++ ;
        if (next_stor == MAXDELAY)
            next_stor -= MAXDELAY;
        }
    else
        {
        printf("There are %d \"@@Ce...\" messages in the ring buffer.\n",\
                            MAXDELAY);
        printf("One correction message was lost.");
        free(ptr->ptr);
        free(ptr);
        return;
        }

    /* Transmit the oldest message if there are more messages
       than the delay length. The delay is premised upon the
       fact that correction messages come approximately one second
       apart.
    */
    while ( (num_waiting - delay) > 0 )
        {
            sio_putb( SGPS_PORT, stor[next_xmit].ptr, stor[next_xmit].len );
            free(stor[next_xmit].ptr);
            next_xmit++ ;
            if (next_xmit == MAXDELAY)
                next_xmit -= MAXDELAY;
            num_waiting-- ;
        };

    free(ptr);
    return;
    };

#endif    /* _GPSDELAY_H */
```

F. GPSFUN.H

```
/*  GPSFUN.H

        A collection of functions to manipulate GPS messages.

            Hex()
            Check_sum()
            Bin_to_ascii()
            Msg_verification()
            Parse_inbuff()
            Master_gps()
            Slave_gps()
*/

#ifndef   _GPSFUN_H
#define   _GPSFUN_H

#include  <stdlib.h>
#include  <string.h>
#include  <stdio.h>
#include  "c:\borlandc\twite\head-c.h"

#ifndef   _GPSTRUCT_H
#include  "c:\borlandc\twite\gpstruct.h"
#endif

FILE     *m_all_file;        /* master file for all data */
FILE     *m_pos_file;        /* master file for pos data */

/* Global matrix for converting ephemeris
*/
UNSIGNED_ONEBYTE  hex_ephemeris[24][3] ;
/* [word][byte] : 24 words (3 bytes per word) corresponding to
    words 3-10 of subframes 1-3 of the satellite nav message.
*/


/**************************************************************
------------------------------------------------------------
    Hex

        This procedure converts an ASCII value to its hex
        equivalent.

        Input  is a single ONEBYTE.
        Output is a single ONEBYTE.
*/
```

```
ONEBYTE Hex(ONEBYTE ASCIIvalue)
    {
    /* 0 is 0x30(48) ->   9 is 0x39(57)
      A is 0x41(65) ->   F is 0x46(70)
      a is 0x61(97) ->   f is 0x66(102)
    */

    if ((ASCIIvalue) < 0x40)
        return((ASCIIvalue) - 0x30);
    else
        if ((ASCIIvalue) < 0x60)
            return((ASCIIvalue) - 0x37);
    else
        return((ASCIIvalue) - 0x57);
    };

/*************************************************************
--------------------------------------------------------------
    Check_sum()

        Calculates the checksum of a message in a buffer of type
        ONEBYTE. Since the message is assumed to be in the Motorola
        Binary Proprietary format, all but the last three bytes
        are used in computing the checksum.

        Input  is the pointer of type PANDL.
        Output is the checksum of type ONEBYTE.
*/

ONEBYTE   Check_sum( PANDL *msg )
    {
    ONEBYTE   chk_sum = 0;
    int  i;

    /* Compute the Exclusive-Or of all but the last three bytes
    */
    for ( i = 0; i < (msg->len - 3); i++ )
        chk_sum ^= msg->ptr[i];
    return chk_sum;
    };

/*************************************************************
--------------------------------------------------------------
    Bin_to_ascii()

        Converts a message from binary to its hexidecimal
    ASCII equivalent. All bytes of the input message are
    converted to ASCII except the first n values as specified
    in the integer input argument. (This was incorporated to
    permit the use of this function with the Motorola GPS
    proprietary message format where the first four bytes are
    already in ASCII.)

    Input  is a pointer of type PANDL and the interger value
           of the number of bytes to skip before conversion.
    Output is a pointer of type PANDL to the new buffer
           with the ASCII converted message.
*/

PANDL   *Bin_to_ascii( PANDL *bin_msg, int skip )
    {
    int          i, j, index, tmplen = 0 ;
    ONEBYTE        nibble[2], *ptr_tool;
    PANDL          *ascii_msg;

    tmplen = bin_msg->len * 2 - skip;
    ptr_tool = bin_msg->ptr;

    /* Create new memory for the converted message.
    */
    ascii_msg = malloc( sizeof( PANDL ));
    ascii_msg->ptr = calloc( tmplen, sizeof(ONEBYTE) );
    ascii_msg->len = tmplen;

    /* Copy any initial values to the new buffer that don't need conversion.
    */
    memcpy(ascii_msg->ptr,ptr_tool,skip);

    for ( i = skip; i < tmplen; i++ )
        {
        /* Split the byte into its upper and lower nibbles.
        */
        nibble[0] = ( ptr_tool[i] >> 4 ) & 0x0F;    /* upper */
        nibble[1] = ( ptr_tool[i] & 0x0F );          /* lower */
        for ( j = 0; j < 2; j++ )
            {
            /* Convert the nibbles to ASCII.
            */
            if ( nibble[j] < 10 )
                nibble[j] += 0x30;      /* nibble+0x30=0x30:0x39 */
            else
                nibble[j] += 0x37;      /* nibble+0x37=0x41:0x46 */
            };

        /* Compute the index  and insert into the ascii_msg buffer
        */
        index = 2 * i - skip;
        ascii_msg->ptr[ index    ] = nibble[0];
        ascii_msg->ptr[ index + 1] = nibble[1];
        };

    return ascii_msg;
    };
/*****************************************************************
-------------------------------------------------------------
    Msg_verification()

        Specifically used to verify the integrity of the
        Motorola GPS proprietary message format.

        Input  are a pointer of type PANDL and an integer designating
            whether the message was received from the GPS receiver (Rx)
            or is to be transmitted to the GPS receiver (Tx).
        Output is the integer flag indicating whether or not the
            message passed verification. True means it passed.
*/

int  Msg_verification( PANDL *msg, int msg_direction )
    {
    int      i, len = -1, array_col;
    int      status = TRUE;

    ONEBYTE      byte0  = msg->ptr[0];        /* Msg prefix @  byte */
    ONEBYTE      byte1  = msg->ptr[1];        /* Msg prefix @  byte */
    ONEBYTE      byte2  = msg->ptr[2];        /* Msg id first  byte */
    ONEBYTE      byte3  = msg->ptr[3];        /* Msg id second byte */
    ONEBYTE      byteCS = msg->ptr[ msg->len-3 ]; /* Check Sum  byte */
    ONEBYTE      byteCR = msg->ptr[ msg->len-2 ]; /* Carr Retn  byte */
    ONEBYTE      byteLF = msg->ptr[ msg->len-1 ]; /* Line feed  byte */

    /* Determine if message verification is for Tx or Rx because the
      number of byte in the message is different for a Tx message
      than for an Rx message.
    */
    if ( msg_direction )
        array_col = 0;          /* Tx message: to GPS receiver */
    else
        array_col = 3;          /* Rx message: fm GPS receiver */

    /* Message ID existence verification and expected length.
    */
    if ( byte2 = = 'A' )
        for ( i = 0; i < 30; i+ + )
            {
            if ( byte3 = = A_array[i][array_col] )
                {
                if ( msg_direction )
                 len = A_array[i][1];        /* Msg size to PVT-6 */
                else
                 len = A_array[i][2];        /* Msg size fm PVT-6 */
                break;
                };
        };
if ( byte2 = = 'B' )
    for ( i = 0; i < 11; i+ + )
        {
        if ( byte3 = = B_array[i][array_col] )
            {
            if ( msg_direction )
              len = B_array[i][1],       /* Msg size to PVT-6 */
            else
              len = B_array[i][2];       /* Msg size fm PVT-6 */
            break;
            };
        };
if ( byte2 = = 'C' )
    for ( i = 0; i < 11; i+ + )
        {
        if ( byte3 = = C_array[i][array_col] )
            {
            if ( msg_direction )
              len = C_array[i][1];       /* Msg size to PVT-6 */
            else
              len = C_array[i][2];       /* Msg size fm PVT-6 */
            break;
            };
        };
if ( byte2 != 'A' && byte2 != 'B' && byte2 != 'C' )
    {
    status = FALSE;
    return status;
    };

/* Message length verification. ("Cj" message is a special case.
   See GPSINIT.H.)
*/
if ( len  != msg->len && byte2 != 'C' && byte3 != 'j' )
    {
    status = FALSE;
    return status;
    };

/* Message check sum verification.
*/
if ( byteCS != Check_sum( msg ) )
    {
    status = FALSE;
    return status;
    };

/* Message begins with @@ prefix.
*/
if ( byte0 != '@' || byte1 != '@' )
```

```
        {
        status = FALSE;
        return status;
        };

    /* Message ends with CR-LF.
    */
    if ( byteCR != 0x0D || byteLF != 0x0A )
        {
        status = FALSE;
        return status;
        };
    return status;
    };

/**********************************************************************
-------------------------------------------------------------
    Parse_inbuff()

        This function accepts a pointer of type PANDL to the
        contents extracted from the serial port Rx buffer.    As
        this function repeatedly parses each succeeding message
        from this buffer, the buffer's length decreases until
        the only remaining information is a partial message.
        The pointer is a static pointer and will not lose its
        information upon completion of this function so it will
        be rejoined with the rest of itself when the next grab
        from the Rx buffer occurs.

        Inputs are a pointer of type PANDL and a pointer to a flag
            indicating when the only remaining contents  constitute
            a partial message.
        Output is a pointer to a new buffer of type PANDL containing
            the next complete single message in the buffer. Also,
            the inbuff pointer passed into Parse_inbuff has been
            altered.
*/

PANDL    *Parse_inbuff( PANDL *start_of_next_message, int *partial_msg_flag )
    {
    int          i = 0, j = 0;
    int      msg_start_found_before = FALSE;
    PANDL        *parsed_msg;
    ONEBYTE        *ptr_tool;

    parsed_msg = malloc( sizeof( PANDL ) );

    ptr_tool = start_of_next_message->ptr;

    /* CASE #1: A complete message has been found. The pointer
       tool points to the beginning of a msg (indicated by "@@")
```

```
   and "@@" has been found once before. Move the place keeper
   pointer "start_of_next_msg->ptr" and adjust the length
   "start_of_next_msg->len". Copy the "found" message to a
   new buffer and return its location in a pointer of type
   PANDL.
*/

for ( i = 0; i < start_of_next_message->len; i++ )
    {
    if ( ptr_tool[i] == '@' && ptr_tool[i + 1] == '@' )
        {
        if ( msg_start_found_before )
            {
            parsed_msg->len = i - j;
            parsed_msg->ptr = calloc(parsed_msg->len, sizeof(ONEBYTE));
            memcpy(parsed_msg->ptr, ptr_tool, parsed_msg->len);

            start_of_next_message->ptr += i;
            start_of_next_message->len -= i;

            *partial_msg_flag = FALSE;
            return parsed_msg;
            };
        msg_start_found_before = TRUE;
        j = i;
        };
    };
/* The start of a second message "@@" was not found. The contents
   are either exactly a complete message or they constitute a
   partial message. Thus, the Parse_inbuff function is
   given another chance otherwise the "partial-msg_flag" is
   returned and the partial message contents will be prepended
   to the next buffer brought into Parse_inbuff.
*/

/* CASE #2: There is exactly one message in the buffer.
*/
if ( ptr_tool[0]      == '@'  && \
     ptr_tool[1]      == '@'  && \
     ptr_tool[ start_of_next_message->len - 2 ] == 0x0D && \
     ptr_tool[ start_of_next_message->len - 1 ] == 0x0A    )
     {
       parsed_msg->len = i - j;
       parsed_msg->ptr = calloc(parsed_msg->len, sizeof(ONEBYTE));
       memcpy(parsed_msg->ptr, ptr_tool, parsed_msg->len);

       start_of_next_message->ptr = NULL;
       start_of_next_message->len = 0;

       *partial_msg_flag = FALSE;
     }
    /* CASE #3: There is only a partial message left in the buffer.
    */
    else
         {
           *partial_msg_flag = TRUE;
         };
    return parsed_msg;
    };

/*****************************************************************
*****************************************************************
    Master_gps()

         For use with a Motorola GPS receiver configured in the
         master station mode. It builds two binary files on disk:
         One records all output messages and the other records just
         position messages. The function returns a pointer of type
         PANDL that points to a buffer with psuedorange correction
         messages ready for transmission to the slave receiver.

         Input  is nothing.
         Output is a pointer of type PANDL.
*/

PANDL *Master_gps(void)
    {
    static    PANDL        mxsbuff;
    static    ONEBYTE   mxs[300];     /* buffer for partial msg */
    static  ONEBYTE          mstart = TRUE ;
            ONEBYTE          *tail;
            PANDL          *inbuff, *tmpbuff, *parse_tool, *ben;
            int          *partial_msg_flag;

    partial_msg_flag = malloc( sizeof( ONEBYTE ) );
    *partial_msg_flag = TRUE;

    inbuff      = malloc( sizeof( PANDL ) );
    tmpbuff     = malloc( sizeof( PANDL ) );
    parse_tool = malloc( sizeof( PANDL ) );
    ben         = malloc( sizeof( PANDL ) );

    /* Determine amount of info in Rx buffer and import to tmpbuff.
    */
    tmpbuff->len = (ONEBYTE) sio_lqueue( MGPS_PORT );
    tmpbuff->ptr = calloc( tmpbuff->len, sizeof(ONEBYTE) );
    sio_read( MGPS_PORT, tmpbuff->ptr, tmpbuff->len );

    /* Initialize the excess buffer that is to contain the partial msg.
    */
    if ( mstart )
         {        /* This "if" statement is invoked    */
    mxsbuff.len = 0;       /* only the first time the program  */
    mstart = FALSE;       /* is run. Note the scope of mstart.*/
    mxsbuff.ptr = mxs;
    };

/* Append tmpbuff messages to the partial message that is in mxsbuff.
*/
inbuff->len = tmpbuff->len + mxsbuff.len;
inbuff->ptr = calloc( inbuff->len, sizeof(ONEBYTE) );
memcpy( inbuff->ptr, mxsbuff.ptr, mxsbuff.len );
tail = inbuff->ptr + mxsbuff.len;
memcpy( tail, tmpbuff->ptr, tmpbuff->len );

/* Free tmpbuff memory and reset partial message buffer length to zero.
*/
free( tmpbuff->ptr );
free( tmpbuff );
mxsbuff.ptr = mxs;
mxsbuff.len = 0;

/* Initialize "ben" for use in collecting pseudorange correction messages.
*/
ben = inbuff;
ben->len = 0;

while ( TRUE )
    {
    /* Get a full message from inbuff.
    */
    parse_tool = Parse_inbuff( inbuff, partial_msg_flag );
    if ( *partial_msg_flag )
        {
        free( parse_tool->ptr );
        break;
        };

    /* Verify the received message validity and indicate when
      the message fails verification.
    */
    if ( Msg_verification( parse_tool, RX ) != TRUE )
        {
        printf("Message from master receiver fails verification!\n");
        free( parse_tool->ptr );
        continue;
        };

    /* Store only psuedorange corrections in old inbuff buffer for
      return to the calling routine.
    */
    if (parse_tool->ptr[2] = = 'C' && parse_tool->ptr[3] = = 'e' )
        memcpy(ben->ptr + ben->len, parse_tool->ptr, parse_tool->len);
    /* Write all messages to this file.
    */
    m_all_file = fopen( "mall.txt", "ab" );
    fwrite(parse_tool->ptr,sizeof(ONEBYTE),parse_tool->len,m_all_file);
    fclose( m_all_file );

    /* Write only position messages to this file.
    */
    if (parse_tool->ptr[2] = = 'B' && parse_tool->ptr[3] = = 'a' )
      {
      m_pos_file = fopen( "mpos.txt", "ab" );
      fwrite(parse_tool->ptr,sizeof(ONEBYTE),parse_tool->len,m_pos_file);
      fclose( m_pos_file );
      };

    free( parse_tool->ptr );
    }; /* end while */

  /* Store any partial message in the excess buffer.
  */
  if ( inbuff->len > 0 )
    {
    memcpy(mxsbuff.ptr, inbuff->ptr, inbuff->len);
    mxsbuff.len = inbuff->len;
    };

  free( inbuff     );
  free( parse_tool );
  free( partial_msg_flag );

  return ben;
  };

/*****************************************************************
------------------------------------------------------------
  Slave_gps()

    For use with a Motorola GPS receiver in the slave
    station mode. It converts five data messages into
    scaled and formatted data:
            Position/Status/Data message
            xDOP Table Status message
            Ephemeris Data Output message
            Satellite Range Data Output message
            Position/Status/Data Extension message
    In addition, these and all other output messages are
    returned to the calling routine for transmission to
    the master receiver for post-processing.

    Input  is a pointer of type PANDL with any input messages.
    Output is a pointer of type PANDL with all output messages.
```

```
*/

PANDL *Slave_gps( PANDL *input)
    {
    static   PANDL        sxsbuff;
    static   ONEBYTE   sxs[300];     /* buffer for partial msg */
    static   ONEBYTE   sstart = TRUE ;
            ONEBYTE   *tail;
            PANDL        *inbuff, *tmpbuff, *parse_tool, *ben;
            int      *partial_msg_flag, i;

    /* Deliver any input messages to the GPS receiver.
    */
    sio_putb( SGPS_PORT, input->ptr, input->len );
    free( input->ptr );
    free( input );

    /* Create flag pointer and working pointers of type PANDL.
    */
    partial_msg_flag = malloc( sizeof(ONEBYTE) );
    *partial_msg_flag = TRUE;                    /* Used in Parse_inbuff */

    inbuff      = malloc( sizeof( PANDL ) );
    tmpbuff     = malloc( sizeof( PANDL ) );
    parse_tool = malloc( sizeof( PANDL ) );
    ben         = malloc( sizeof( PANDL ) );

    /* Determine amount of info in Rx buffer and import to tmpbuff.
    */
    tmpbuff->len = (ONEBYTE) sio_iqueue( SGPS_PORT );
    tmpbuff->ptr = calloc( tmpbuff->len, sizeof(ONEBYTE) );
    sio_read( SGPS_PORT, tmpbuff->ptr, tmpbuff->len );

    /* Initialize the excess buffer that is to contain the partial msg.
    */
    if ( sstart )
        {         /* This "if" statement is invoked   */
        sxsbuff.len = 0;        /* only the first time the program  */
        sstart = FALSE;      /* is run. Note the scope of sstart.*/
        sxsbuff.ptr = sxs;
        };

    /* Append this info in tmpbuff to excess msg from last read.
    */
    inbuff->len = tmpbuff->len + sxsbuff.len;
    inbuff->ptr = calloc( inbuff->len, sizeof(ONEBYTE) );
    memcpy( inbuff->ptr, sxsbuff.ptr, sxsbuff.len );
    tail = inbuff->ptr + sxsbuff.len;
    memcpy( tail, tmpbuff->ptr, tmpbuff->len );

    /* Free tmpbuff memory and reset partial message buffer length to zero.
```

```
*/
free( tmpbuff->ptr );
free( tmpbuff );
sxsbuff.ptr = sxs;
sxsbuff.len = 0;

/* Initialize "ben" for use in returning all messages to the calling
  environment.
*/
ben = inbuff;

while ( TRUE )
    {
    /* Get a full message from inbuff.
    */
    parse_tool = Parse_inbuff( inbuff, partial_msg_flag );
    if ( *partial_msg_flag )
        {
        free( parse_tool->ptr );
        break;
        };

    /* Verify the received message validity.
    */
    if ( Msg_verification( parse_tool, RX ) != TRUE )
        {
        free( parse_tool->ptr );
        continue;
        };

    /* Convert the binary msg to GPS data structure for use by cpu.
    */
    switch (parse_tool->ptr[3])
        {
        case 'a':                /* @@Ba Position/Status/Data    */
            /* Insert conversion here */
            break;
        case 'c':                /* @@Bc xDOP Table Status       */
            /* Insert conversion here */
            break;
        case 'f':                /* @@Bf Ephemeris Data           */
             for ( i = 5 ; i < EPH_NUM_WORDS+5 ; i+=3 )
                {
                hex_ephemeris[i][0] = parse_tool->ptr[i + 0] ;
                hex_ephemeris[i][1] = parse_tool->ptr[i + 1] ;
                hex_ephemeris[i][2] = parse_tool->ptr[i + 2] ;
                }
            /* Insert conversion here */
            break;
        case 'g':                /* @@Bg Satellite Range          */
            /* Insert conversion here */
```

```
                break;
            case 'k':                   /* @@Bk Position/Status/Data Ext  */
                /* Insert conversion here */
                break;
            default:
                break;
            };

        free( parse_tool->ptr );
        }; /* end while */

    /* Store any partial message in the excess buffer.
    */
    if ( inbuff->len > 0 )
        {
        memcpy(sxsbuff.ptr, inbuff->ptr, inbuff->len);
        sxsbuff.len = inbuff->len;
        };
    free( inbuff    );
    free( parse_tool );
    free( partial_msg_flag );

    return ben;
    };

#endif   /* _GPSFUN_H */
```

G. GPSINIT.H

```
/*  GPSINIT.H

            This header file contains the Motorola Proprietary
            Binary Message Format characteristcs for GPS
            receivers. The first column contains the second
            letter of the message ID of a message prepared for
            input to the GPS receiver. The fourth column contains
            the second letter of the message ID of a message
            received from the GPS receiver. For all messages sent
            to the GPS receiver except one, there is a message reply.
               The second column contains the final length of a
            message prepared to be sent to the GPS receiver. The
            third column is the length of a message received from
            the GPS receiver with that particular message ID.
               Note: All message lengths can be expressed with
            one byte except one: The response message Cj is 294 bytes.
            A unique dummy size equal to 1 byte has been used to
            indicate the length.
*/

#ifndef  _GPSINIT_H
#define  _GPSINIT_H

#ifndef _GPSDEFIN_H
#include  "c:\borlandc\twite\gpsdefin.h"
#endif

ONEBYTE          A_array[30][4]={{      'a', 10,  10,  'a'  },
                   {   'b', 10,  10,  'b'  },
                   {   'c', 11,  11,  'c'  },
                   {   'd', 11,  11,  'd'  },
                   {   'e', 11,  11,  'e'  },
                   {   'f', 12,  15,  'f'  },
                   {   'g', 8,   8,   'g'  },
                   {   'h', 8,   8,   'h'  },
                   {   'i', 9,   9,   'i'  },
                   {   'j', 8,   8,   'j'  },
                   {   'k', 9,   9,   'k'  },
                   {   'l', 9,   9,   'l'  },
                   {   'm', 12,  12,  'm'  },
                   {   'n', 8,   8,   'n'  },
                   {   'o', 8,   25,  'p'  },
                   {   'p', 25,  25,  'p'  },
                   {   'q', 8,   8,   'q'  },
                   {   'r', 8,   8,   'r'  },
                   {   's', 20,  20,  's'  },
                   {   't', 8,   8,   't'  },
                        {   'u', 12,  12,  'u'  },
                        {   'v', 8,   8,   'v'  },
                        {   'w', 8,   8,   'w'  },
                        {   'x', 9,   9,   'x'  },
                        {   'y', 11,  11,  'y'  },
                        {   'z', 11,  11,  'z'  },
                        {   'A', 8,   8,   'A'  },
                        {   'B', 8,   8,   'B'  },
                        {   'C', 9,   9,   'C'  },
                        {   'D', 9,   9,   'D'  }};

ONEBYTE         B_array[11][4]={{       'a', 8,   68,  'a'  },
                        {   'b', 8,   92,  'b'  },
                        {   'c', 8,   82,  'c'  },
                        {   'd', 8,   23,  'd'  },
                        {   'e', 8,   0,   '-'  },/* Be->Cb */
                        {   'i', 8,   80,  'f'  },
                        {   'g', 8,   122,'g'   },
                        {   'h', 8,   0,   '-'  },/* Bh->Ce */
                        {   'j', 7,   8,   'j'  },
                        {   'f', 80,  0,   '-'  },/* Bf->Cc */
                        {   'k', 8,   69,  'k'  }};

ONEBYTE         C_array[11][4]= {{ 'a', 7,   9,   'a'  },
                        {   'b', 33,  9,   'h'  },
                        {   'd', 27,  171,'d'   },
                        {   'e', 52,  7,   'k'  },
                        {   'f', 7,   7,   'f'  },
                        {   'g', 8,   8,   'g'  },
                        {   'i', 8,   0,   'i'  },
                        {   'j', 7,   1,   'j'  }, /* Cj resp msg 294 byte */
                        {   '-', 0,   33,  'b'  }, /* Be->Cb */
                        {   '-', 0,   52,  'e'  }, /* Bh->Ce */
                        {   '-', 0,   80,  'c'  }};/* Bf->Cc */

#endif   /* _GPSINIT_H */
```

H.   GPSPORTS.H

```
/*  GPSPORTS.H

        Initialize_master_gps()
        Initialize_slave_gps()
        Open_ports()
        Close_ports()

        Both initialization functions need to be greatly improved
        to provide an easy to understand and use interface with
        the user.
*/

#ifndef  _GPSPORTS_H
#define  _GPSPORTS_H

#include  <stdio.h>
#include  "c:\borlandc\twite\head-c.h"

/*************************************************************
*************************************************************
    Initialize_master_gps()

        This function presents the current configuration
        settings for the master GPS receiver. The user is
        offered the opportunity to change any of the parameters.
        In addition, the user is requested to indicate which if
        any output files he will want. For those files, the default
        file name is presented and made available for modification
        or change.

        Input  is the serial port number through which the master
            GPS communicates.
        Output is nothing.
*/

void  Initialize_master_gps(int mgps)
    {
    char  s1[20] = {'@','@','B','a',1,22,13,10};
    char  s2[20] = {'@','@','B','h',1,43,13,10};
    sio_flush(mgps,2);
    sio_putb(mgps,s1,8);
    sio_putb(mgps,s2,8);
    printf("Master GPS receiver initialized.\n");
    return;
    };

/*************************************************************
```

```
-----------------------------------------------------------------
    Initialize_slave_gps()

        This function presents the current configuration
        settings for the slave GPS receiver. The user is
        offered the opportunity to change any of the parameters.
        In addition, the user is requested to indicate which if
        any output files he will want. For those files, the default
        file name is presented and made available for modification
        or change.

        Input  is the serial port number through which the slave
            GPS communicates.
        Output is nothing.

*/

void  Initialize_slave_gps(int sgps)
    {
    char  s1[20]={'@','@','B','a',1,22,13,10};
    sio_flush(sgps,2);
    sio_putb(sgps,s1,8);
    printf("Slave GPS receiver initialized.\n");
    return;
    };

/***************************************************************
-----------------------------------------------------------------
    Open_ports()

            This function initializes the two serial ports
        on the PCD-744 to be used by the master and slave GPS
        receivers. Before initialization, it either loads the
        drivers for the standard and PCD-744 serial ports or
        verifies that they are loaded. It then configures the
        Baud rating, word length, parity, and stop bits as well
        as the modem and flow control. All commands that begin
        with "sio_..." are PCD-744 commands and may be referenced
        in the PCLS-802 PC-ComLIB Serial Communication Programming
        Library.
            The PCD-744 RS-232 8 port intelligent IO card is
        made by ADVANTECH in Sunnyvale, Ca. (408) 245-6678.
            Input is the slave GPS port number.
            Output is nothing.
*/

void  Open_ports( int mgps, int sgps )
    {
    int  BAUD     = 12;              /* 9600 (p.12 PC-ComLIB) */
    int  IOMODE   = (0x03|0x00|0x00);          /* 8-N-1 (p. 12)  */
    int  MODMODE  = 0x00;                 /* DTR and RTS off (p.26)*/
int   HWMODE    = 0x00;             /* HW and SW flow ctrl off
                     (p.33)*/
int   n;

    n = sio_ioctl( mgps, BAUD, IOMODE );
        if (n != 0)
        {
            printf("Opening comport %d ioctl error.\n",mgps);
            abort();
        };
    n = sio_lctrl( mgps, MODMODE );
        if (n != 0)
        {
            printf("Opening comport %d lctrl error.\n",mgps);
            abort();
        };
    n = sio_flowctrl( mgps, HWMODE );
        if (n != 0)
        {
            printf("Opening comport %d flowctrl error.\n",mgps);
            abort();
        };
    n = sio_open( mgps );
        if (n != 0)
        {
            printf("Opening comport %d error.\n",mgps);
            abort();
        };


    n = sio_ioctl( sgps, BAUD, IOMODE );
        if (n != 0)
        {
            printf("Opening comport %d ioctl error.\n",sgps);
            abort();
        };
    n = sio_lctrl( sgps, MODMODE );
        if (n != 0)
        {
            printf("Opening comport %d lctrl error.\n",sgps);
            abort();
        };
    n = sio_flowctrl( sgps, HWMODE );
        if (n != 0)
        {
            printf("Opening comport %d flowctrl error.\n",sgps);
            abort();
        };
    n = sio_open( sgps );
        if (n != 0)
        {
                printf("Opening comport %d error.\n",sgps);
                abort();
            };
    printf("Opened ports %d (Master GPS) and %d (Slave GPS).\n", \
                            mgps, sgps);
    return;
    };

/*************************************************************
-------------------------------------------------------------
    Close_ports()

        This function closes and resets all serial ports for
        the PCD-744 Serial IO card. See the Open_ports function
        for more information on the commands used herein.

        Inputs are the master and slave GPS port numbers.
        Output is nothing.
*/

void  Close_ports( int mgps, int sgps )
  {
  printf("Closing all ports\n");
  sio_close( sgps );
  sio_close( mgps );
  sio_reset();
  return;
  };

#endif    /* _GPSPORTS_H */
```

I.   GPSTRUCT.H

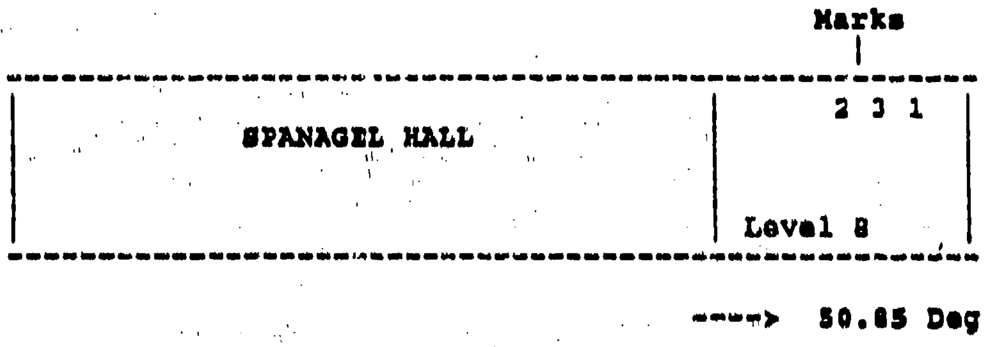
```
/*  GPSTRUCT.H
*/

#ifndef  _GPSTRUCT_H
#define  _GPSTRUCT_H

#ifndef  _GPSINIT_H
#include "c:\borlandc\twite\gpsinit.h"
#endif

typedef      struct{            /* Pointer AND Len */
         ONEBYTE   *ptr;
         int      len;
            }PANDL;

/* Struct and global data definitions
*/

struct T_FLOAT_EPHEMERIS {
  UNSIGNED_ONEBYTE iode;

  /* Field name abbreviations were extracted from ICD-GPS-200.
  */
  double toe;           /* Time of ephemeris.                    */
  double m0;             /* Mean anomaly at refernce time.          */
  double delta_n;         /* Mean motion difference from computed value.  */
  double e;            /* Eccentricity.                   */
  double sqrt_a;         /* Square root of semi-major axis.          */
  double omega_0;          /* Longitude of ascending node of orbit plane
                        at weekly epoch.                   */
  double i0;            /* Inclination angle at reference time.       */
  double w;             /* Argument of perigee.                 */
  double omega_dot;       /* Rate of right ascension.               */
  double i_dot;          /* Rate of inclination angle.             */
  double cuc;            /* Amplitude of the cosine harmonic correction
                        term to the argument of latitude.       */
  double cus;            /* Amplitude of the sine harmonic correction
                       term to the argument of latitude.       */
  double crc;            /* Amplitude of the cosine harmonic correction
                        term to the orbit radius.          */
  double crs;            /* Amplitude of the sine harmonic correction
                        term to the orbit radius.          */
  double cic;            /* Amplitude of the cosine harmonic correction
                        term to the angle of inclination.       */
  double cis;            /* Amplitude of the sine harmonic correction
                        term to the angle of inclination.       */
  double tgd;            /* Estimated group delay differential.        */
  double toc;          /* Clock data reference time.              */
  double af2;          /* Polynomial coefficient (SV clock correction) */
  double af1;          /* Polynomial coefficient (SV clock correction) */
  double af0;          /* Polynomial coefficient (SV clock correction) */
  double n;           /* Corrected mean motion.                   */
  double e_sqrt;        /* Square root of eccentricity^2 subtracted
                       from 1.                              */
  double wk1;           /* Omega_dot - WE.                          */
  double wk0;           /* Omega_0 - WE * toe.                       */
  double sin_w;         /* Sine w.                             */
  double cos_w;          /* Cosine w.                            */
  double a;           /* Semi-major axis.                        */
  double sqrta_e_fconst; /* Square root of semi-major axis *
                  eccentricity * FCONST.                  */
} float_ephemeris ;

struct T_RECEIVER_CHANNELS {
  UNSIGNED_ONEBYTE svid;
  UNSIGNED_ONEBYTE mode;
  UNSIGNED_ONEBYTE strength;
  UNSIGNED_ONEBYTE flags;
} ;

struct T_GEODETIC  {
  TWOBYTE          degrees ;
  UNSIGNED_TWOBYTE minutes ;
  double           seconds ;
} ;

struct T_POS_CHAN_STATUS {
  UNSIGNED_ONEBYTE      month;
  UNSIGNED_ONEBYTE      day;
  UNSIGNED_TWOBYTE       year;
  UNSIGNED_ONEBYTE      hours;
  UNSIGNED_ONEBYTE      minutes;
  double                seconds;
  struct T_GEODETIC      latitude;
  struct T_GEODETIC      longitude;
  double               datum_height;  /* meters */
  double               msl_height;    /* meters */
  double               velocity;     /* m/sec */
  double               heading;      /* degrees */
  double                 current_dop;
  UNSIGNED_ONEBYTE      dop_type;
  UNSIGNED_ONEBYTE      visible_sats;
  UNSIGNED_ONEBYTE      sats_tracked;
  struct T_RECEIVER_CHANNELS     channel[NUM_CHANNELS] ;
  UNSIGNED_ONEBYTE      rcvr_status;
};

#endif   /* _GPSTRUCT_H */
```

J. HEAD-C.H

```
/*  moxa-c.h  ver 1.00  10/11/1991
 *  Definitions for MOXA Serial I/O Controller ioctrl
 */

/*  BAUD rate setting   */
#define B50        0x00
#define B75        0x01
#define B110       0x02
#define B134       0x03
#define B150       0x04
#define B300       0x05
#define B600       0x06
#define B1200      0x07
#define B1800      0x08
#define B2400      0x09
#define B4800      0x0A
#define B7200      0x0B
#define B9600      0x0C
#define B19200     0x0D
#define B38400     0x0E
#define B57600     0x0F

/*  MODE setting       */
#define BIT_5      0x00         /* Word length define    */
#define BIT_6      0x01
#define BIT_7      0x02
#define BIT_8      0x03

#define STOP_1     0x00         /* Stop bits define */
#define STOP_2     0x04

#define P_EVEN     0x18         /* Parity define    */
#define P_ODD      0x08
#define P_SPC      0x38
#define P_MRK      0x28
#define P_NONE     0x00

/*  MODEM CONTROL setting   */
#define C_DTR      0x01
#define C_RTS      0x02

/*  MODEM LINE STATUS   */
#define S_CTS      0x01
#define S_DSR      0x02
#define S_RI       0x04
#define S_CD       0x08

/*  function declaration    */
```

```
long sio_iqueue(int);
long sio_ifree(int);
long sio_oqueue(int);
long sio_ofree(int);
```

SPANAGEL GEODETIC MARKS
WGS-84



Spanagel Hall runs SW to NE. The marks are on the 8 th level near the North-East cornor of the building.

Marks 1 and 2 are standard survey disks, marked SPA-1 and SPA-2. SPA-3 is a square of thin aluminum. All marks are attached to the roof under a rail.

Poles are mounted on the rail above the marks. The height to the marks are 11.83 m (WGS-84 elipsoid).

			Mark	Pole
SPA-1	36 35.70375	121 52.47777	11.83	14.08
SPA-2	36 35.70180	121 52.48074	11.83	14.10
SPA-3	36 35.70277	121 52.47926	11.83	14.09

Table IX WGS 84 SPANAGAL HALL EIGHTH FLOOR GEODETIC MARKS.

Location	Item	degrees	degrees minutes	degrees minutes seconds	meters (above geoid)
Spa-1	Lat	36.59506253 °N	36° 35.70375'	36° 35' 42.2251"	
2321	Lon	121.87462956 °W	121° 52.47777'	121° 52' 28.6664"	
	Alt				11.83
Spa-2	Lat	36.59503006 °N	36° 35.70180'	36° 35' 42.1082"	
2322	Lon	121.87467903 °W	121° 52.48074'	121° 52' 28.8445"	
	Alt				11.83
Spa-3	Lat	36.59504622 °N	36° 36.70277'	36° 35' 42.1664"	
2323	Lon	121.87465439 °W	121° 52.47926'	121° 52' 28.7558"	
	Alt				11.83

LIST OF REFERENCES

Advantech, *PCLS-802 PC-ComLIB Serial Communication Programming Library,* Revision 3.0, December 1992.

Anderson, John D., Jr., *Introduction to Flight*, New York, McGraw-Hill Book Company, 1978.

Applied Research Laboratories, *GPS Seminar*, presented to the Naval Security Group, University of Texas, Austin, 26-28 January 1993.

Beck, G. E., *Navigation Systems: A Survey of Modern Electronic Aids,* Van Nostrand Reinhold, 1971.

Bowditch, Nathaniel, *American Practical Navigator: An Epitome of Navigation,* Defense Mapping Agency Hydrographic/Topographic Center, Vol I, 1984.

Carey, Susan, *Sextants in Space Can Change the World,* Wall Street Journal, v. CXXIX, no 13, p. B1.

Chan, Ling, "GPS World Receiver Survey," *GPS World*, v. 4, no. 1, pp. 52-65, January 1993.

Dunlap, G. D., and Shufeldt, H. H., *Dutton's Navigation and Piloting,* 12th ed., United States Naval Institute, 1969.

Hobbs, Richard R., *Marine Navigation 2: Celestial and Electronic*, Naval Institute Press,1983.

Kayton, Myron, *Navigation: Land, Sea, Air and Space*, New York, IEEE Press, 1990.

Kremer, G. T., and others, *The Effect of Selective Availability on Differential GPS Corrections,* Navigation, Journal of the ION, v. 37, No. 1, Spring 1990.

Leick, Alfred, *GPS Satellite Surveying*, New York, John Wiley & Sons, 1990.

Lin, Ching-Fang, *Modern Navigation, Guidance, and Control Processing*, pp. 215-231, 445-464, Englewood Cliffs, N.J., Prentice-Hall, Inc., 1991.

Logsdon, Tom, *The Navstar Global Positioning System*, Van Nostrand Reinhold, 1992.

Motorola, *GPS Receiver Technical Reference Manual,* Revision 4.0, 1992.

U.S. Naval Oceanographic Office, *H.O. Pubs 117A and 117B Radio Navigation Aids,* Government Printing Office, Washington, DC, 1972.

Rockwell International Corporation, *Interface Control Document ICD-GPS-200 (Revision B),* 1987.

Tattersfield, Donald, *Orbits for Amateurs with a Microcomputer, New York,* John Wiley & Sons, 1984.

Wells, D.E., and others, *Guide to GPS Positioning*, 2nd ed., Fredericton, N.B., Canada, Canadian GPS Associates, 1987.

Wilkes, Owen and Gleditsch, Nils Petter, *Loran-C and Omega: A Study of the Military Importance of Radio Navigation Aids,* Oslo, Norwegian University Press, 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey CA 93943-5101	2
3.	Professor Michael Shields, Code EC/SL Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	4
4.	Professor Isaac Kaminer, Code AA/KA Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5121	2
5.	Professor Rick Howard, Code AA/HB Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5106	1
6.	Professor Robert McGhee, Code CS/MZ Department of Computer Science Naval Postgraduate School Monterey, California 93943-55118	1
7.	Lieutenant Commander Eric Twite 32 Bahama Bend Coronado, California 92118	2